

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-187638

(43) 公開日 平成10年(1998) 7月21日

(51) Int.Cl.⁸

G 0 6 F 15/16

識別記号

3 8 0

F I

G 0 6 F 15/16

3 8 0 Z

審査請求 未請求 請求項の数15 O L (全 42 頁)

(21) 出願番号 特願平9-75254

(22) 出願日 平成9年(1997) 3月27日

(31) 優先権主張番号 特願平8-285398

(32) 優先日 平8(1996)10月28日

(33) 優先権主張国 日本 (J P)

(71) 出願人 000006013

三菱電機株式会社

東京都千代田区丸の内二丁目2番3号

(72) 発明者 細川 武彦

東京都千代田区丸の内二丁目2番3号 三

菱電機株式会社内

(72) 発明者 鶴 薫

東京都千代田区丸の内二丁目2番3号 三

菱電機株式会社内

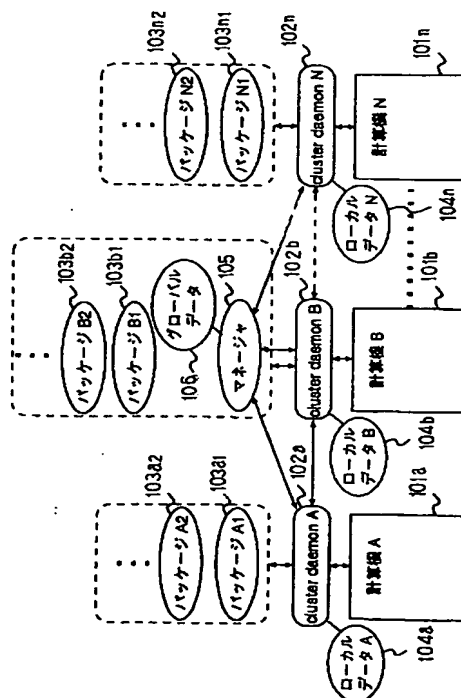
(74) 代理人 弁理士 宮田 金雄 (外2名)

(54) 【発明の名称】 クラスタ制御システム

(57) 【要約】

【課題】 クラスタシステムを監視制御し、障害が発生した場合に当計算機上で動作していたプログラムをクラスタ内の他の計算機に移行して実行させることを目的とする。

【解決手段】 クラスタを構成する各計算機上でクラスタデーモンが実行されており、各パッケージを起動すると同時に、実行計算機上のリソースを監視制御し、そのデータを各計算機上にローカルデータとして保持する。マネージャは、各計算機上のクラスタデーモンと通信を行ない、クラスタシステム全体の監視制御のためのグローバルデータを保持している。また、マネージャは、クラスタシステムのパッケージの1つであり、マネージャやマネージャを実行している計算機に障害が発生した場合、クラスタデーモンにより他の計算機上で再起動される。



【特許請求の範囲】

【請求項1】 クラスタシステムを構成する計算機群上のある計算機に障害が発生した場合に該計算機上で動作中のパッケージプログラムを他の計算機で実行させるクラスタシステムにおいて、
 クラスタを構成する各計算機は、
 アプリケーションや各種のサービスを提供するパッケージプログラムと、
 計算機間で通信を行いリソースを監視制御するクラスタデーモンと、
 上記監視結果をローカルデータとして記憶するローカルデータ記憶手段を備え、
 クラスタシステム内のうち1台の計算機は、上記パッケージプログラム、クラスタデーモン、ローカルデータ記憶手段に加えて、
 各計算機上のローカルデータから収集されて、いずれの計算機からも参照可能なグローバルデータ記憶手段と、
 上記グローバルデータ記憶手段および各計算機上のクラスタデーモンと通信を行い、クラスタシステム全体の監視制御を行うマネージャを搭載し、
 該マネージャが搭載されている計算機で障害が発生した場合にはクラスタ内の他の計算機上で再起動させるようにしたことを特徴とするクラスタ制御システム。

【請求項2】 クラスタシステムを構成する計算機群上のある計算機に障害が発生した場合に該計算機上で動作中のパッケージプログラムを他の計算機で実行させるクラスタシステムにおいて、
 クラスタを構成する各計算機は、
 アプリケーションや各種のサービスを提供するパッケージプログラムと、
 計算機間で通信を行いリソースを監視制御するクラスタデーモンと、
 自計算機上のクラスタデーモンおよびマネージャと通信を行うエージェントと、
 上記監視結果をローカルデータとして記憶するローカルデータ記憶手段を備え、
 クラスタシステム内のうち1台の計算機は、上記クラスタデーモン、エージェント、ローカルデータ記憶手段に加えて、
 各計算機上のローカルデータから収集されて、いずれの計算機からも参照可能なグローバルデータ記憶手段と、
 上記グローバルデータ記憶手段および各計算機上のエージェントと通信を行い、クラスタシステム全体の監視制御を行うマネージャを搭載し、
 該マネージャが搭載されている計算機に障害が発生した場合にクラスタ内の他の計算機上で再起動させるようにしたことを特徴とするクラスタ制御システム。

【請求項3】 クラスタシステムを構成する計算機群上のある計算機に障害が発生した場合に該計算機上で動作していたパッケージプログラムを他の計算機で実行させ

るクラスタシステムにおいて、
 クラスタを構成する各計算機は、
 アプリケーションや各種のサービスを提供するパッケージプログラムと、
 自計算機上のパッケージプログラムおよび計算機間で通信を行いリソースを監視制御するクラスタデーモンと、
 自計算機上のクラスタデーモン、各計算機上のエージェント間、およびグローバルデータ記憶手段と通信を行うエージェントと、
 上記監視結果をローカルデータとして記憶するローカルデータ記憶手段を備え、
 クラスタシステム内のうち1台の計算機は、上記クラスタデーモン、エージェント、ローカルデータ記憶手段に加えて、
 各計算機上のローカルデータから収集されて、いずれの計算機からも参照可能なグローバルデータ記憶手段を備え、
 上記各計算機上のエージェントが直接にグローバルデータ記憶手段、およびエージェント間で通信を行うようにしたことを特徴とするクラスタ制御システム。

【請求項4】 クラスタシステムを構成する計算機群上のある計算機に障害が発生した場合に該計算機上で動作していたパッケージプログラムを他の計算機で実行させるクラスタシステムにおいて、
 クラスタを構成する各計算機は、
 アプリケーションや各種のサービスを提供するパッケージプログラムと、
 自計算機上のパッケージプログラムおよび各計算機間で通信を行いリソースを監視制御するクラスタデーモンと、
 自計算機上のクラスタデーモン、各計算機上のエージェント間、およびグローバルデータと通信を行うエージェントと、
 上記監視結果をローカルデータとして記憶するローカルデータ記憶手段を備え、
 クラスタシステム内のうち1台の計算機は、上記クラスタデーモン、エージェント、ローカルデータ記憶手段に加えて、
 各計算機上のローカルデータから収集されて、いずれの計算機からも参照可能なグローバルデータ記憶手段と、
 自計算機上のエージェントおよびクラスタデーモンと通信を行うマネージャを備え、
 上記各計算機上のエージェントが直接にグローバルデータ記憶手段、およびエージェント間で通信を行うようにしたことを特徴とするクラスタ制御システム。

【請求項5】 上記マネージャはクラスタシステムを構成する計算機群のリソース状態変化時の処理を記述したリソース設定ファイルと、
 リソース設定ファイルの定義に従い、リソースの状態に変化があった場合にリソース制御処理を行なう自動制御

機構を備えたことを特徴とする請求項1又は請求項2又はまたは請求項4記載のクラスタ制御システム。

【請求項6】 前記リソース設定ファイルにはパッケージプログラム間の相関関係や実行に関する優先順位情報を定義し、

自動制御機構は、該定義情報に基づいて各計算機上のパッケージプログラムを動作させるようにしたことを特徴とする請求項5記載のクラスタ制御システム。

【請求項7】 上記マネージャはパッケージプログラムに対して、運転、待機、試験を含む運転動作モードを付加し、

該モードに従ってパッケージプログラムの動作制御の管理を行なうモード管理機構を備えたことを特徴とする請求項1、請求項2、請求項4、請求項5のいずれかに記載のクラスタ制御システム。

【請求項8】 上記マネージャは、クラスタシステム内で起きたリソースの状態変化に関するログを収集するログ管理機構を備えたことを特徴とする請求項1、請求項2、請求項4乃至請求項7のいずれかに記載のクラスタ制御システム。

【請求項9】 クラスタ制御システムを構成する複数の計算機のうちの1つの計算機に障害が発生した場合に、上記障害が発生した計算機で運転中のアプリケーションや各種のサービスを提供するパッケージプログラムを他の計算機で運転させるクラスタ制御システムにおいて、上記複数の計算機はそれぞれ、自己の計算機の障害及び回復を監視するとともに、上記パッケージプログラムの起動及び運転を制御するクラスターデーモンを備え、上記複数の計算機のうちの第1の計算機は、上記パッケージプログラムである第1のパッケージプログラムを運転し、

上記複数の計算機のうちの第2の計算機は、上記第1のパッケージプログラムと同じアプリケーションやサービスを提供する第2のパッケージプログラムを起動状態で待機させ、

上記複数の計算機のうち1つの計算機は、上記クラスターデーモンに加えて、

上記複数の計算機のそれぞれのクラスターデーモンから監視の結果を受け取るとともに、上記クラスターデーモンを制御して、上記第1の計算機に障害が発生した場合に、上記第2の計算機に上記第2のパッケージプログラムを運転させるとともに、上記第1の計算機が障害から回復した場合には、上記第1の計算機に上記第1のパッケージプログラムを起動状態で待機させるマネージャを備えたことを特徴とするクラスタ制御システム。

【請求項10】 上記第1のパッケージプログラムの出力若しくは上記第2のパッケージプログラムの出力のいずれかを選択して出力する出力制御手段を有し、

上記第2の計算機は、上記第2のパッケージプログラムを起動状態で待機させる代わりに、上記第2のバッケー

ジプログラムを運転し、

上記マネージャに代えて、上記複数の計算機のそれぞれのクラスターデーモンから監視の結果を受け取るとともに、上記クラスターデーモンを制御して、上記第1のパッケージプログラムの出力が上記出力制御手段から出力されているときに上記第1の計算機で障害が発生した場合、上記第1のパッケージプログラムの出力に代えて上記第2のパッケージプログラムの出力を上記出力制御手段から出力させ、上記第2の計算機で障害が発生するまで上記出力制御手段に上記第2のパッケージプログラムの出力を継続して出力させ、上記第1のパッケージプログラムが運転を再開し上記第2の計算機で障害が発生した場合に、上記第2のパッケージプログラムの出力に代えて、上記第1のパッケージプログラムの出力を上記出力制御手段から出力させるマネージャを備えたことを特徴とする請求項9に記載のクラスタ制御システム。

【請求項11】 上記マネージャに代えて、上記複数の計算機のそれぞれのクラスターデーモンから監視の結果を受け取るとともに、上記クラスターデーモンを制御して、上記第1の計算機に障害が発生した場合に、上記第2の計算機に上記第2のパッケージプログラムを運転させるとともに、上記複数の計算機のうちの第3の計算機に上記第1のパッケージプログラムを起動状態で待機させるマネージャを備えたことを特徴とする請求項9に記載のクラスタ制御システム。

【請求項12】 上記複数の計算機のそれぞれで起動されるパッケージプログラムのそれぞれの優先順位を記憶する管理テーブルを備え、

上記マネージャに代えて、上記複数の計算機のそれぞれのクラスターデーモンから監視の結果を受け取るとともに、上記クラスターデーモンを制御して、上記第1の計算機に障害が発生した場合に、上記管理テーブルから上記第1の計算機で運転されていた上記パッケージプログラムよりも優先順位の低いパッケージプログラムを検索し、この優先順位の低いパッケージプログラムの運転を停止させるとともに、上記優先順位の低いパッケージプログラムを運転していた計算機で上記第1の計算機で運転されていたパッケージプログラムを起動させるマネージャを備えたことを特徴とする請求項9に記載のクラスタ制御システム。

【請求項13】 クラスタシステムを構成する複数の計算機のうちの1つの計算機に障害が発生した場合に、上記障害が発生した計算機で運転中のアプリケーションや各種のサービスを提供するパッケージプログラムを他の計算機で運転させるクラスタシステムにおいて、上記複数の計算機はそれぞれ、自己の計算機の障害及び回復を監視するとともに、上記パッケージプログラムの起動及び運転を制御するクラスターデーモンを備え、

上記複数の計算機のうち1つの計算機は、上記クラスターデーモンに加えて、

上記複数の計算機のそれぞれで起動されるバックエンドプログラムのそれぞれの優先順位及び上記複数の計算機のそれぞれの負荷を記憶する管理テーブルと、

上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記複数の計算機のうちの第1の計算機の負荷があらかじめ定められた負荷よりも大きくなった場合に、上記管理テーブルを参照し、上記第1の計算機で運転しているバックエンドプログラムのうちの優先順位の低いバックエンドプログラムの運転を停止させ、停止させたバックエンドプログラムを上記複数の計算機のうちの負荷があらかじめ定められた負荷よりも小さい計算機で起動させるマネージャと、を備えたことを特徴とするクラスタ制御システム。

【請求項14】 上記管理テーブルに代えて、上記複数の計算機上で起動されるバックエンドプログラムのそれぞれの優先順位を記憶する管理テーブルを備え、上記クラスタデーモンは、自己の計算機のリソースを監視し、

上記マネージャに代えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記クラスタデーモンにより監視されたリソースに変化が生じた場合に、上記優先順位に基づいて、上記複数の計算機のそれぞれで運転されているバックエンドプログラムのそれぞれにリソースを割り当て直すマネージャを備えたことを特徴とする請求項13に記載のクラスタ制御システム。

【請求項15】 複数の計算機のそれぞれによって並列に運転される複数の上記バックエンドプログラムを1つのグループとするグループ名を記憶する管理テーブルを備え、

上記マネージャに代えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記複数の計算機のうちの第1の計算機に障害が発生した場合に、上記管理テーブルから、上記複数の計算機のうちの計算機であって上記第1の計算機上で運転されていたバックエンドプログラムと同じグループのバックエンドプログラムを運転していない計算機を検索し、検索された計算機で上記第1の計算機が運転していたバックエンドプログラムを起動させ、運転させるマネージャを備えたことを特徴とする請求項13に記載のクラスタ制御システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、クラスタシステムを監視制御し、ある計算機上で障害が発生した場合に該計算機上で動作していたバックエンドプログラムをクラスタを構成する他の計算機に移行して実行するクラスタ制御システムに関するものである。

【0002】

【従来の技術】従来クラスタと呼ばれる技術には大きく分けて、CPUが主記憶を共有する密結合型クラスタと、計算機がLANや共有ディスクなどを用いてデータを共有する疎結合型クラスタの2つが存在するが、ここでの記述は後者の疎結合型クラスタについてである。図50は、従来のクラスタシステムの構成例を示す説明図である。図において、計算機A～N(101a～101n)はクラスタを構成する複数の計算機である。各計算機上では、cluster daemon A～N(102a～102n)が実行されており、各cluster daemonにより各バックエンドプログラムA1～N2(103a1～103n2)が起動される。ここで、バックエンドプログラムとは、アプリケーションプログラムやサービスプログラム等の総称である。各cluster daemonは、実行している計算機上のリソース(CPU, LAN, Disk, バックエンドプログラム)が提供する各種サービス、ネットワークアドレスなどを監視/制御し、そのデータを各計算機上にローカルデータA～N(104a～104n)として保持している。

【0003】次に、クラスタシステムの動作について、図51に基づいて説明する。計算機A(101a)上で必要なリソースA(2401a)がなくなった場合、cluster daemon A(102a)は計算機A(101a)を停止させる。計算機A(101a)が停止した場合、他の計算機N(101n)上のcluster daemon N(102n)がこれを検知し、計算機A(101a)上で実行していたバックエンドプログラムA(103a)を他の計算機N(101n)で実行する。このようにして、バックエンドプログラムは、クラスタ内のどこかの計算機上で実行されることになる。また、各バックエンドプログラム毎にネットワークアドレスを割り当てることにより、バックエンドプログラムが提供するサービスを利用する場合において、ユーザはバックエンドプログラムがクラスタ内のどの計算機上で実行されているかを意識する必要がなかった。また、分散しているリソースの状態を集中して監視制御する方式として、特開平5-75628号「ネットワーク資源監視システム」、特開平5-134902号「分散コンピューティングシステムでの稼働情報管理方式」、特開平6-223020号「ネットワーク管理システムおよびオブジェクトの管理方法」などがあった。しかしながら、これらの方式は管理用の計算機や管理用のプロセス(マネージャ)を用いることにより実現しており、管理用の計算機や管理用のプロセス(マネージャ)に障害が発生した場合については考慮されていなかった。

【0004】

【発明が解決しようとする課題】従来のクラスタシステムは以上のようにして構成されているので、システム全体の監視および制御を行うプログラムを作成する場合に

においては、データが各計算機に分散されていて、クラスタ内の全ての計算機と通信しなければならず、プログラムの作成が困難であるという問題点があった。また、分散しているリソース状態を集中監視制御する方式においては、システム全体を監視制御する計算機やプロセスに障害が発生した場合に監視機能が全く停止するという問題点があった。また、各種パッケージプログラム間の相関関係や優先順位などの定義が行なえなかったため、多重系などの他のシステムからの移行が困難であるという問題点があった。また、パッケージの再起動に時間がかかり、回復に時間がかかるという問題があった。また、システムが回復後のパッケージの切り替え処理に時間がかかり、並列処理を行うパッケージが並列処理を行えないため、回復後にシステムのパフォーマンスが落ちるという問題があった。

【0005】この発明は上記のような問題点を解消するためになされたもので、クラスタシステム全体の監視および制御を行うプログラムの作成を容易にするとともに、他システムからの移行を可能とし、高速に動作するクラスタ制御システムを提供することを目的とする。

【0006】

【課題を解決するための手段】第1の発明に係わるクラスタ制御システムは、クラスタシステムを構成する計算機群上のある計算機に障害が発生した場合に該計算機上で動作中のパッケージプログラムを他の計算機で実行させるクラスタシステムにおいて、クラスタを構成する各計算機が、アプリケーションや各種のサービスを提供するパッケージプログラムと、計算機間で通信を行いリソースを監視制御するクラスターデーモンと、監視結果をローカルデータとして記憶するローカルデータ記憶手段を備え、クラスタシステム内のうち1台の計算機は、上記パッケージプログラム、クラスターデーモン、ローカルデータ記憶手段に加えて、各計算機上のローカルデータから収集されて、いずれの計算機からも参照可能なグローバルデータ記憶手段と、グローバルデータ記憶手段および各計算機上のクラスターデーモンと通信を行い、クラスタシステム全体の監視制御を行うマネージャを搭載し、マネージャが搭載されている計算機で障害が発生した場合にはクラスタ内の他の計算機上で再起動させるようにしたものである。

【0007】第2の発明に係わるクラスタ制御システムは、クラスタシステムを構成する計算機群上のある計算機に障害が発生した場合に計算機上で動作中のパッケージプログラムを他の計算機で実行させるクラスタシステムにおいて、クラスタを構成する各計算機が、アプリケーションや各種のサービスを提供するパッケージプログラムと、計算機間で通信を行いリソースを監視制御するクラスターデーモンと、自計算機上のクラスターデーモンおよびマネージャと通信を行うエージェントと、監視結果をローカルデータとして記憶するローカルデータ記憶手

段を備え、クラスタシステム内のうち1台の計算機は、クラスターデーモン、エージェント、ローカルデータ記憶手段に加えて、各計算機上のローカルデータから収集されて、いずれの計算機からも参照可能なグローバルデータ記憶手段と、グローバルデータ記憶手段および各計算機上のエージェントと通信を行い、クラスタシステム全体の監視制御を行うマネージャを搭載し、マネージャが搭載されている計算機に障害が発生した場合にクラスタ内の他の計算機上で再起動させるようにしたものである。

【0008】第3の発明に係わるクラスタ制御システムは、クラスタシステムを構成する計算機群上のある計算機に障害が発生した場合に計算機上で動作していたパッケージプログラムを他の計算機で実行させるクラスタシステムにおいて、クラスタを構成する各計算機が、アプリケーションや各種のサービスを提供するパッケージプログラムと、自計算機上のパッケージプログラムおよび計算機間で通信を行いリソースを監視制御するクラスターデーモンと、自計算機上のクラスターデーモン、各計算機上のエージェント間、およびグローバルデータ記憶手段と通信を行うエージェントと、監視結果をローカルデータとして記憶するローカルデータ記憶手段を備え、クラスタシステム内のうち1台の計算機は、上記クラスターデーモン、エージェント、ローカルデータ記憶手段に加えて、各計算機上のローカルデータから収集されて、いずれの計算機からも参照可能なグローバルデータ記憶手段を備え、各計算機上のエージェントが直接にグローバルデータ記憶手段、およびエージェント間で通信を行うようにしたものである。

【0009】第4の発明に係わるクラスタ制御システムは、クラスタシステムを構成する計算機群上のある計算機に障害が発生した場合に計算機上で動作していたパッケージプログラムを他の計算機で実行させるクラスタシステムにおいて、クラスタを構成する各計算機が、アプリケーションや各種のサービスを提供するパッケージプログラムと、自計算機上のパッケージプログラムおよび各計算機間で通信を行いリソースを監視制御するクラスターデーモンと、自計算機上のクラスターデーモン、各計算機上のエージェント間、およびグローバルデータと通信を行うエージェントと、監視結果をローカルデータとして記憶するローカルデータ記憶手段を備え、クラスタシステム内のうち1台の計算機は、上記クラスターデーモン、エージェント、ローカルデータ記憶手段に加えて、各計算機上のローカルデータから収集されて、いずれの計算機からも参照可能なグローバルデータ記憶手段と、自計算機上のエージェントおよびクラスターデーモンと通信を行うマネージャを備え、各計算機上のエージェントが直接にグローバルデータ記憶手段、およびエージェント間で通信を行うようにしたものである。

【0010】第5の発明は、第1または第2または第4

の発明に係わるクラスタ制御システムにおいて、マネージャはクラスタシステムを構成する計算機群のリソース状態変化時の処理を記述したリソース設定ファイルと、リソース設定ファイルの定義に従い、リソースの状態に変化があった場合にリソース制御処理を行なう自動制御機構を備えるようにしたものである。

【0011】第6の発明は、第5の発明に係わるクラスタ制御システムにおいて、リソース設定ファイルにはパッケージプログラム間の相関関係や実行に関する優先順位情報を定義し、自動制御機構は、該定義情報に基づいて各計算機上のパッケージプログラムを動作させるよう

にしたものである。
【0012】第7の発明は、第1または第2または第4または第5のいずれかの発明に係わるクラスタ制御システムにおいて、マネージャはパッケージプログラムに対して、運転、待機、試験を含む運転動作モードを付加し、該モードに従ってパッケージプログラムの動作制御の管理を行なうモード管理機構を備えるようにしたものである。

【0013】第8の発明は、第1、第2、第4乃至第7のいずれかの発明に係わるクラスタ制御システムにおいて、マネージャは、クラスタシステム内で起きたリソースの状態変化に関するログを収集するログ管理機構を備えるようにしたものである。

【0014】また、第9の発明は、クラスタ制御システムを構成する複数の計算機のうちの1つの計算機に障害が発生した場合に、上記障害が発生した計算機で運転中のアプリケーションや各種のサービスを提供するパッケージプログラムを他の計算機で運転させるクラスタ制御システムにおいて、上記複数の計算機はそれぞれ、自己の計算機の障害及び回復を監視するとともに、上記パッケージプログラムの起動及び運転を制御するクラスタデーモンを備え、上記複数の計算機のうちの第1の計算機は、上記パッケージプログラムである第1のパッケージプログラムを運転し、上記複数の計算機のうちの第2の計算機は、上記第1のパッケージプログラムと同じアプリケーションやサービスを提供する第2のパッケージプログラムを起動状態で待機させ、上記複数の計算機のうちの1つの計算機は、上記クラスタデーモンに加えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記第1の計算機に障害が発生した場合に、上記第2の計算機に上記第2のパッケージプログラムを運転させるとともに、上記第1の計算機が障害から回復した場合には、上記第1の計算機に上記第1のパッケージプログラムを起動状態で待機させるマネージャを備えたものである。

【0015】また、第10の発明は、上記第1のパッケージプログラムの出力若しくは上記第2のパッケージプログラムの出力のいずれかを選択して出力する出力制御

手段を有し、上記第2の計算機は、上記第2のパッケージプログラムを起動状態で待機させる代わりに、上記第2のパッケージプログラムを運転し、上記マネージャに代えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記第1のパッケージプログラムの出力が上記出力制御手段から出力されているときに上記第1の計算機で障害が発生した場合、上記第1のパッケージプログラムの出力に代えて上記第2のパッケージプログラムの出力を上記出力制御手段から出力させ、上記第2の計算機で障害が発生するまで上記出力制御手段に上記第2のパッケージプログラムの出力を継続して出力させ、上記第1のパッケージプログラムが運転を再開し上記第2の計算機で障害が発生した場合に、上記第2のパッケージプログラムの出力に代えて、上記第1のパッケージプログラムの出力を上記出力制御手段から出力させるマネージャを備えたものである。

【0016】また、第11の発明は、上記マネージャに代えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記第1の計算機に障害が発生した場合に、上記第2の計算機に上記第2のパッケージプログラムを運転させるとともに、上記複数の計算機のうちの第3の計算機に上記第1のパッケージプログラムを起動状態で待機させるマネージャを備えたものである。

【0017】また、第12の発明は、上記複数の計算機のそれぞれで起動されるパッケージプログラムのそれぞれの優先順位を記憶する管理テーブルを備え、上記マネージャに代えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記第1の計算機に障害が発生した場合に、上記管理テーブルから上記第1の計算機で運転されていた上記パッケージプログラムよりも優先順位の低いパッケージプログラムを検索し、この優先順位の低いパッケージプログラムの運転を停止させるとともに、上記優先順位の低いパッケージプログラムを運転していた計算機で上記第1の計算機で運転されていたパッケージプログラムを起動させるマネージャを備えたものである。

【0018】また、第13の発明は、クラスタシステムを構成する複数の計算機のうちの1つの計算機に障害が発生した場合に、上記障害が発生した計算機で運転中のアプリケーションや各種のサービスを提供するパッケージプログラムを他の計算機で運転させるクラスタシステムにおいて、上記複数の計算機はそれぞれ、自己の計算機の障害及び回復を監視するとともに、上記パッケージプログラムの起動及び運転を制御するクラスタデーモンを備え、上記複数の計算機のうちの1つの計算機は、上記クラスタデーモンに加えて、上記複数の計算機のそれぞれで起動されるパッケージプログラムのそれぞれの優先

順位及び上記複数の計算機のそれぞれの負荷を記憶する管理テーブルと、上記複数の計算機のそれぞれのクラスターデーモンから監視の結果を受け取るとともに、上記クラスターデーモンを制御して、上記複数の計算機のうちの第1の計算機の負荷があらかじめ定められた負荷よりも大きくなった場合に、上記管理テーブルを参照し、上記第1の計算機で運転しているパッケージプログラムのうちの優先順位の低いパッケージプログラムの運転を停止させ、停止させたパッケージプログラムを上記複数の計算機のうちの負荷があらかじめ定められた負荷よりも小さい計算機で起動させるマネージャと、を備えたものである。

【0019】また、第14の発明は、上記管理テーブルに代えて、上記複数の計算機上で起動されるパッケージプログラムのそれぞれの優先順位を記憶する管理テーブルを備え、上記クラスターデーモンは、自己の計算機のリソースを監視し、上記マネージャに代えて、上記複数の計算機のそれぞれのクラスターデーモンから監視の結果を受け取るとともに、上記クラスターデーモンを制御して、上記クラスターデーモンにより監視されたリソースに変化が生じた場合に、上記優先順位に基づいて、上記複数の計算機のそれぞれで運転されているパッケージプログラムのそれぞれにリソースを割り当て直すマネージャを備えたものである。

【0020】また、第15の発明は、複数の計算機のそれぞれによって並列に運転される複数の上記パッケージプログラムを1つのグループとするグループ名を記憶する管理テーブルを備え、上記マネージャに代えて、上記複数の計算機のそれぞれのクラスターデーモンから監視の結果を受け取るとともに、上記クラスターデーモンを制御して、上記複数の計算機のうちの第1の計算機に障害が発生した場合に、上記管理テーブルから、上記複数の計算機のうちの計算機であって上記第1の計算機上で運転されていたパッケージプログラムと同じグループのパッケージプログラムを運転していない計算機を検索し、検索された計算機で上記第1の計算機が運転していたパッケージプログラムを起動させ、運転させるマネージャを備えたものである。

【0021】

【発明の実施の形態】

実施の形態1. 以下、この発明の第1の実施形態について、図1乃至図5に基づいて説明する。図1において、計算機A~N(101a~101n)はクラスタを構成する複数の計算機である。各計算機上では、cluster daemon A~N(102a~102n)が実行されており、各cluster daemonにより各パッケージプログラムA1~N2(103a1~103n2)が起動される。パッケージプログラムとは、アプリケーションやサービスの総称である。また、各cluster daemonは、実行している計算機上の

リソース(CPU、LAN、Disk、パッケージプログラム、ネットワークアドレスなど)を監視制御し、そのデータを各計算機上にローカルデータA~N(104a~104n)として保持している。マネージャ105は、各計算機上のcluster daemonと通信を行なうことにより、クラスタシステム全体の監視制御を行なうものであり、グローバルデータ106を保持している。グローバルデータ106は、クラスタ内の計算機のどこからでも参照することができるデータであり、共有ディスクや共有メモリ、メモリのレプリケーションなどにより実現されている。また、マネージャ105は、クラスタシステム内のパッケージの1つであり、マネージャやマネージャを実行している計算機に障害が発生した場合、cluster daemonにより他の計算機上で再起動される。

【0022】次に、マネージャの構成を図2に基づいて説明する。図において、要求処理機構201は、ユーザからの要求(D201:リソース状態取得要求、リソース制御要求、通知設定要求)を受信し、要求に応じた処理を行ない(D202, D203, D204)、ユーザに処理結果(D205)を送信する機構である。リソース制御機構202は、要求処理機構201からの要求(D204)を受け、要求の対象となるリソースを保持している計算機をリソース状態DB(203)を参照(D206)することにより決定し、その計算機に対してリソースの制御要求(D207)を送信するものである。リソース状態監視機構204は定期的にクラスタ内の各リソースに対し、リソースを保持している計算機にリソース状態取得要求(D208)を送信し、その回答であるリソース状態(D209)を受信し、リソース状態DB(203)を参照(D210)し、リソースの状態が変化した時にリソース状態変化処理機構205に通知(D211)するものである。リソース状態変化処理機構205は、リソース状態監視機構204からの通知(D211)を受け、リソース状態DB(203)を更新(D212)するとともに、通知設定DB(207)を参照(D213)し、通知設定がされていればリソース状態変化通知機構208に通知(D214)するものである。通知設定機構206は、要求処理機構201からの要求(D202)に従い、通知設定DB(207)を更新(D215)し、リソース状態DB(203)を参照(D216)し、現在の状態をリソース状態変化通知機構208に通知(D217)するものである。リソース状態変化通知機構208は、リソース状態変化処理機構205、または通知設定機構206からの通知(D214, D217)を受け、設定を行なったユーザに対してリソース状態変化通知(D218)を送信するものである。通信制御機構209は、ユーザ、cluster daemonとの送受信を制御するものである。リソース状態DB(203)は、リソースの名前、状態、

保持している計算機などの情報を格納しているデータベースである。通知設定DB(207)は、マネージャと接続しているどのユーザに対して、どのリソースの状態が変化した時に通知を行なうかについての情報を格納しているデータベースである。

【0023】次に、ユーザから要求があった場合のマネージャの処理動作について、図3に基づいて説明する。まず、ステップ301において、ユーザからの要求(D201)を待ち、ステップ302において、ユーザからの要求(D201)の種類を判別する。ユーザからの要求がリソースの状態取得であった場合、ステップ303において要求処理機構201はリソース状態DBを参照(D203)し、要求のあったリソースの状態を取得する。ステップ304において、要求処理機構201がユーザへリソースの状態を送信(D205)し、ステップ301に戻る。一方、ユーザからの要求がリソースの制御であった場合、ステップ305において、リソース制御機構202は要求処理機構201から要求(D204)を受け、リソース状態DBを参照(D206)し、要求のあったリソースの状態を保持している計算機を取得する。ステップ306において、リソース制御機構202はステップ305で取得した計算機へ制御要求(D207)を送信する。ステップ307において、要求処理機構201はステップ306のリソース制御機構202の処理結果をユーザへ送信(D205)し、ステップ301に戻る。ユーザからの要求がリソースの状態変化通知の設定であった場合、ステップ308において、通知設定機構206は通知設定DBを更新(D215)する。ステップ309において、要求処理機構201はリソース制御機構202を起動する(D204)とともにユーザへ処理結果を送信(D205)する。ステップ310において、通知設定機構206はリソース状態DB(203)より現在のリソースの状態を取得(D216)する。ステップ311において、リソース状態変化通知機構208は通知設定機構206からの通知(D217)を受け、ステップ310で取得したリソースの状態をユーザへ通知(D218)し、ステップ301に戻る。ステップ310においてユーザへの通知を行なうのは、現在のリソースの状態(ユーザにとっての初期値)を返すためである。

【0024】次に、マネージャのリソースの監視処理について、図4に基づいて説明する。ステップ401において、ステップ402以降の処理を全リソースについて行なう。ステップ402において、リソース状態監視機構204はリソースの状態をそのリソースを保持している計算機より取得(D208、D209)する。ステップ403において、リソース状態監視機構204はリソース状態DB(203)を参照(D210)し、その状態と取得したリソースの状態が同じであれば、ステップ401に戻り、それ以外ならばステップ404以降の処

理を行なう。ステップ404において、リソース状態変化処理機構205はリソース状態監視機構204から通知(D211)を受け、リソース状態DB(203)の状態を取得したリソースの状態に更新(D212)する。ステップ405において、リソース状態変化処理機構205は通知設定DBを参照(D213)し、そのリソースの状態変化に対するユーザへの通知が登録されていればステップ406において、リソース状態変化通知機構208はリソース状態変化処理機構205から通知(D214)を受け、ユーザにリソースの状態変化を通知(D218)し、それ以外ならば、ステップ401に戻る。ステップ407において、一定時間停止する。

【0025】図5はクラスタシステム内における監視制御用プロセスの構成例を示したものである。図において、マネージャ105と同一の計算機A(101a)上のプロセスA(501a)、クラスタ内の他の計算機B(101b)上のプロセスB(501b)、WSなどのようなクラスタ外の計算機C(502)上のプロセスC(501c)からクラスタ上の全リソースの監視制御を行うことが可能となる。また、各パッケージプロセスはマネージャ105に対してのみアクセスすればよく、マネージャがどの計算機上で動作しているかを意識することなく、クラスタの監視制御を行うことができる。

【0026】実施の形態2. 本発明の第2の実施形態について、図6乃至図11に基づいて説明する。本実施形態は図1のプロセス構成を変更したものであり、cluster daemonが他の計算機からアクセスできない場合や、マネージャおよびネットワークの負荷を軽くしたい場合の構成である。図6は、本実施形態におけるプロセス構成を示す図である。エージェントA~N(601a~601n)は、各計算機上で実行されて、各計算機上のcluster daemonおよびマネージャと通信を行なうものである。尚、その他の要素は、図1において同一番号を付したのものと同一構成要素である。

【0027】図7は、本実施形態におけるマネージャの構成である。図2との相違点は、リソース状態監視機構204を削除し、リソース状態変化処理機構205はエージェントからの通知を受信(D701)するようにしたこと、及びリソース制御機構202はcluster daemonに対してではなくエージェントにリソース制御要求を送信(D702)するようにしたことである。

【0028】図8は、本実施形態におけるエージェントの構成を示す図である。各機構の動作は図2と同様であるが以下の点が異なっている。即ち、通知設定DB(207)、および通知設定機構206が削除され、要求処理機構201に対する要求は、マネージャからのリソース制御要求(D702)だけである。リソース制御機構202は、リソース状態DB(203)を参照せずに、

動作している計算機上のcluster daemonにリソース制御要求(D702)を送信する。リソース状態変化処理機構205は、通知設定DB(207)を参照せずに全てのリソース状態監視機構204からの通知(D211)をリソース状態変化通知機構208に通知(D214)する。リソース状態変化通知機構208は、ユーザに対してではなく、マネージャに対してリソース状態変化通知(D701)を行う。リソース状態DB(801)は、エージェントが実行されている計算機が保持するローカルなリソース状態のみを管理している。通信制御機構209の送信時にマネージャ105が存在しなかった場合、そのデータはキューに保持される。ユーザからの要求があった場合におけるマネージャの処理は、図3と同様である。

【0029】次に、図10に基づいて、マネージャのリソースの監視処理について説明する。ステップ1001において、リソース状態監視機構204はエージェントからのリソース状態変化通知(D701)を待つ。ステップ1002~1004は、図4におけるステップ404~406と同様である。

【0030】次に、エージェントのリソースの監視処理について、図11に基づいて説明する。ステップ1101において、ステップ1102以降の処理を計算機上の全リソースについて行なう。ステップ1102において、リソース状態監視機構204はリソースの状態をcluster daemonより取得(D208、D209)する。ステップ1103において、リソース状態監視機構204はリソース状態DBを参照(D210)し、その状態と取得したリソースの状態が同じであれば、ステップ1101に戻り、それ以外ならばステップ1104以降の処理を行なう。ステップ1104において、リソース状態変化処理機構205はリソース状態監視機構204から通知(D211)を受け、リソース状態DBの状態を取得したリソースの状態に更新(D212)する。ステップ1105において、リソース状態変化通知機構208はリソース状態変化処理機構205から通知(D214)を受け、マネージャにリソースの状態変化を通知(D701)し、ステップ1101に戻る。また、ステップ1106において一定時間停止する。

【0031】更に、図9は図8の変形例を示した図であり、cluster daemonにリソースの状態変化の通知を行なう機能がある場合のエージェントの構成である。各機構の基本動作は図8と同様であるが、リソース状態監視機構204、リソース状態DB(801)を削除し、通知設定機構206を追加した点で異なっている。ここで、通知設定機構206はエージェントの起動またはcluster daemonの起動時に全てのリソースの状態変化を通知するようにcluster daemonに通知設定要求(D901)を送信する

機構である。また、リソース状態変化処理機構205は、cluster daemonからの通知(D902)を受け、リソース状態変化通知機構208に通知(D903)する。

【0032】以上の構成により、cluster daemonが同一計算機上のプロセスとしか通信できない場合においても、同一計算機上のエージェントがcluster daemonと通信し、マネージャはエージェントと通信するため、マネージャによるクラスタの集中監視制御が可能となる。また、各エージェントがリソースの状態を定期的に取得するようにしたので、負荷が分散され、ネットワークを流れるデータが少なくなり、マネージャ、ネットワークの負荷を軽くすることが可能である。

【0033】実施の形態3. 本発明の第3の実施形態について、図12、図13に基づいて説明する。本実施形態は、図6のプロセス構成を変更したものであり、マネージャを削除してエージェント同士が互いに通信し、グローバルデータを参照するようにしたものである。次に、図12に基づいてプロセス構成を説明する。

エージェントA~N(601a~601n)は、各エージェントA~Nが搭載された計算機上で実行されるcluster daemon、他の計算機上のエージェント、およびユーザと通信を行なうものである。グローバルデータ106は、各計算機からアクセス可能であり、各計算機からのアクセスは排他的に行なわれる。エージェントの構成は、図2のマネージャの構成と同様であるが、各機構は以下の点で異なっている。リソース状態DB(203)、通知設定DB(207)に対するアクセスは、各エージェントで排他をとりアトミックに行なわれる。また、各エージェントのリソース状態監視機構204は、各エージェントの実行されている計算機上のリソースのみの監視を行なう。また、リソース制御機構202は、制御するリソースが計算機上に存在すれば、計算機上のcluster daemonに送信し、存在しなければリソース状態DB(203)を参照して、そのリソースを保持している計算機上のエージェントまたは、cluster daemonに送信する。更に、通信制御機構209における各ユーザからの受信、ユーザへの送信は各ユーザまたは各エージェントからの受信、ユーザまたはエージェントへの送信となる。

【0034】図13は、本構成を用いたユーザプロセスの構成例を示す図である。ユーザプロセスA(1301)は、エージェント601a~601nの内のどれか1つのエージェントを選び通信を行なう。ここで、図13(b)のようにエージェントに障害が発生した場合は、他のエージェントを選び再接続する。ユーザプロセスB(1302)は、エージェント601a~601nの内のどれか2つのエージェントを選び通信を行なうものである。この場合には、図13(b)のように片方の

エージェントに障害が発生した場合でも、もう一方のエージェントと通信を行なうことにより、処理を継続することが可能である。以上の構成により、ユーザは、クラスタ内のどれか一つのエージェントと通信することによりクラスタの集中監視制御が可能となる。また、2つのエージェントと通信することにより、エージェントに障害が発生した場合の通信不能時間を短縮することが可能となる。

【0035】実施の形態4. 本発明の第4の実施形態について、図14、図15に基づいて説明する。本実施形態は、図12においてマネージャ105を追加したものである。以下に、図14を用いてプロセス構成を説明する。ここで、エージェントA～N(601a～601n)は、図12に記載のものと同様である。マネージャ105は、ネットワークアドレスのみを持つパッケージプログラムとして構成され、ユーザはマネージャ105にアクセスすることにより、クラスタ内のどれか1つのエージェントに対しアクセスを行うことができる。

【0036】図15は、本構成を用いたユーザプロセスの構成例である。ユーザプロセスA(1501)は、マネージャ105と通信を行なうものである。図15

(b)のようにマネージャ105に障害が発生した場合は、他の計算機上でマネージャ105が再起動され、ユーザプロセスA(1501)は、マネージャ105と再接続される。ユーザプロセスB(1502)は、エージェント601a～601nの内のどれか2つのエージェントを選び通信を行なうものである。図15(b)のように片方のエージェントに障害が発生した場合でも、もう一方のエージェントと通信を行なうことにより、処理を継続することが可能である。以上の構成により、ユーザは実施形態1、実施形態2のようにマネージャに対してのみアクセスを行なうか、または実施形態3のようにクラスタ内のどれか一つのエージェントにアクセスするかの運用形態を適宜選択することが可能となる。また、マネージャに対してのみアクセスを行なう場合、マネージャはネットワークアドレスのみを有するパッケージプログラムとして構成されるので、実施形態1、および2と比べ短い時間でマネージャの移動が行なえるので、マネージャに障害が発生した場合でも通信不能時間を短縮することができる。

【0037】実施の形態5. 本発明の第5の実施形態について、図16、図17に基づいて説明する。本実施形態は図7のマネージャにおいて、マネージャに設定ファイルと自動制御機構を備えたものである。図16は、本実施形態におけるマネージャの構成図である。図において、設定ファイル1601はリソース状態の変化に基づいてシステムの自動制御を行なうためのものであり、リソース名、イベント名、処理の組を記述したものである。また、イベント名、処理中の変数名(attribute)、およびコマンド名(method)はリソー

スの種類により決まるものであり、変数名は“リソース名. 変数名”と表記し、コマンド名は“リソース名→コマンド名”と表記する。自動制御機構1602は、設定ファイル1601を読み込み(D1601)、リソース状態変化処理機構205からの通知(D1602)を受けた時に、リソース状態DB(203)を参照(D1603)し、設定ファイル1601の定義に従い、リソース制御機構202にリソース制御要求(D1604)を送信する機構である。

【0038】次に、図17は、設定ファイルの例を示した図である。図において、pkg1、pkg2はリソース名であり、リソースの種類はパッケージである。1～3行目はコメントである。5～9行目はpkg1が停止した時にpkg2が起動されていなければpkg2を起動するための定義である。11～15行目はpkg1が起動した時にpkg2がpkg1と同じ計算機上で起動されていればpkg2を停止するための定義である。17～21行目はpkg2が起動した時にpkg1がpkg2と同じ計算機上で起動されていればpkg2を停止する(pkg2の起動を中止する)ための定義である。以上の定義では、pkg1とpkg2が同じ計算機上では動作しないようにし、同じ計算機上で動作しようとした場合にはpkg1を優先するようなシステムとなる。本設定ファイルではパッケージに対する制御を例としたが、もちろん他のリソースに対する制御を行なっても良い。また、本実施形態では実施形態2のマネージャについて説明したが、実施形態1、実施形態4に記載のマネージャに対して実施しても良い。

【0039】実施の形態6. 本発明の第6の実施形態について、図18に基づいて説明する。本実施形態は図17の設定ファイルの変形例に関するもので、パッケージ間の相関関係や優先順位の設定を可能としたものである。設定ファイルにパッケージ名とパッケージの実行可能条件の組の記述を加えたものであり、自動制御機構は設定ファイルの設定に従いリソース制御機構により、リソースの制御を可能としたものである。パッケージの実行可能条件はパッケージ名、|、&、!の列であり、パッケージ名は同じ計算機上でそのパッケージが起動していることを条件とし、|、&、!は条件の論理和、論理積、論理否定を表す。また、記述した順番によりパッケージの優先順位は定義されるものとする。

【0040】図18は、設定ファイルの例である。1行目はコメントである。2行目は、pkg1は同じ計算機上でpkg2が動作してはいけなことを定義している。3行目は、pkg2とpkg1が同じ計算機上で動作することを禁止するための定義である。4行目は、pkg3はpkg1またはpkg2が、同じ計算機上で動作しなければならないことを定義している。以上の定義によれば、pkg3は同じ計算機上でpkg1またはpkg2のどちらか片方のみが動作している計算機上で

実行されるようなシステムとなる。

【0041】実施の形態7. 本発明の第7の実施形態について、図19乃至図21に基づいて説明する。本実施形態は、図7のマネージャ構成にモード管理機構を備えたものである。同じパッケージプログラムが複数の計算機上で実行可能な場合に、各パッケージプログラムに“運転”、“待機”、“試験”などのモードの情報を付加し、モードの情報に基づいてパッケージプログラムの動作を変更するようなマネージャを実現する。

【0042】図19は、本実施形態におけるマネージャの構成を示す図である。モード管理機構1901はリソース状態変化処理機構205からの通知(D1901)を受け、リソース状態DB203を参照(D1902)し、図20に示すような動作を行うようにリソース制御機構202に対しリソース制御要求(D1903)を送信する機構である。

【0043】図20は、パッケージプログラムのモード状態遷移を示す図である。モードが停止のパッケージプログラムは“運転”、“待機”、“試験”のモードを指定して起動することによって各状態に遷移し、各状態からは、停止または、障害発生時にモードが“停止”の状態に遷移する。また、モードが“待機”中のパッケージプログラムは、“運転”中の他のパッケージプログラムが停止した場合に、“運転”のモードに状態遷移する。また、モード状態が“運転”のパッケージは、常にクラスタ内で1つであるように制御される。

【0044】図21は、2台の計算機が動作する場合の運用例を示す図である。図において、パッケージA(2101, 2102)は同一のパッケージであり、各計算機上で実行されている。図21(a)のように計算機1上のパッケージA(2101)のモードを“運転”とし、計算機2上のパッケージA(2102)のモードを“待機”としていた場合、計算機1に障害が発生した時(図21(b))、計算機2上のパッケージA(2102)のモードを“運転”とし処理を継続する(図21(c))。計算機1が再起動された時には、クラスタ内に運転のパッケージA(2102)があるため、計算機1上では、パッケージA(2101)のモードを待機として再起動する(図21(d))。以上のような構成をとることにより、パッケージの再起動を行なうよりも短時間で切替を行なうことが可能となる。本実施形態ではモードの状態として、“運転”、“待機”、“試験”を定義したが、他の状態を定義してもよい。また、本実施形態では実施形態2のマネージャを用いているが、勿論、実施形態1、実施形態4、実施形態5のマネージャを使用しても良い。

【0045】実施の形態8. 本発明の第8の実施形態について、図22に基づいて説明する。本実施形態は、図7のマネージャ構成にログ管理機構を備えたものである。本実施形態におけるマネージャの構成を図22に示

す。図において、ログ管理機構2201は、リソース状態変化処理機構205からリソースの状態変化の通知(D2201)を受けログDB(2202)を更新(D2202)する。この時、ログのデータは時系列に並ぶように制御する。また、要求処理機構201からの要求(D2203)を受け、ログDB(2202)を参照(D2204)し、ログデータを返す。ログDB(2202)には、クラスタ内の全リソースの全イベントと発生した時間が保存される。以上の構成により、ユーザはマネージャに対してアクセスするだけでクラスタ内で発生した全イベントの情報を取得することが可能となる。また、本実施形態では、実施形態2のマネージャについて説明したが、勿論、実施形態1、実施形態4、実施形態5、実施形態7のマネージャを使用しても良い。

【0046】実施の形態9. 本発明の第9の実施の形態について、図23～図26に基づいて説明する。本実施形態は、図1に示したクラスタシステムのマネージャ105の代わりに用いられ、各パッケージをモードで管理する機能を有するマネージャを備えたものである。なお、以降、計算機A101a～計算機N101nのうちどれかを特定しないときは計算機101といい、クラスタデーモンA102a～N102nを特定しないときはクラスタデーモン102、パッケージ103a1～103n2のうちどれかを特定しないときはパッケージ103という。

【0047】本実施形態におけるマネージャの構成を図23に示す。図23において、通知処理機構2301は、クラスタデーモン102からのリソース状態変化通知を受信し(D2301)、モード管理機構2302に受け取ったリソース状態変化通知を送る(D2302)機構である。リソース状態変化通知(D2301)は、計算機101が停止したときなど、システム全体のリソースに変化があったときに、当該リソースを管理するクラスタデーモン102から送信される通知である。モード管理機構2302は、通知処理機構2301からのリソース状態変化通知(D2302)を受け、管理テーブル2303を参照/更新し、モード制御機構2304にモード制御要求(D2304)を送信する機構である。モード制御機構2304は、モード管理機構2302からのモード制御要求(D2304)を受信し、クラスタデーモン102またはパッケージ103に対してモード制御要求を送信する(D2305)機構である。

【0048】図24は管理テーブル2303の記憶内容の例を示す。管理テーブル2303には、パッケージ名とモードの状態、例えば「運転」、「待機」、「停止」等が保持されている。図25は、図23のマネージャ105の処理を説明するフローチャートである。

【0049】次に、図25に基づいて、マネージャ105の動作について説明する。ステップ2501において、モード管理機構2302は管理テーブル2303の

初期化を行う。次に、ステップ2502において、通知処理機構2301はクラスタデーモン102からのリソース状態変化通知(D2301)を待ち、受信したらモード管理機構2302へリソース状態変化通知(D2302)を送信する。続いて、ステップ2503において、モード管理機構2302はリソース状態変化通知(D2302)の種類を判別する。リソース状態変化通知(D2302)が計算機101の起動である場合にはステップ2507へ進み、計算機101の停止である場合にはステップ2504へ進む。計算機の停止であった場合、ステップ2504において、モード管理機構2302は、管理テーブル2303を更新し、停止した計算機101上で運転または待機していたすべてのパッケージ103のモードを「停止」に書き換える。

【0050】次に、ステップ2505において、モードが「待機」のパッケージ103があるかを調べる。このとき、複数種類のパッケージ103が計算機上で実行されている場合には、停止したパッケージ103と同種類のパッケージ103であってモードが「待機」であるパッケージ103があるかを調べる。そのために、図24に示した管理テーブル2303に、パッケージ103が実行されている計算機の情報及びパッケージ103の種類の情報を追加してもよい。ステップ2505で、モードが「待機」のパッケージ103がなかった場合にはステップ2502に戻る。ここで、モードが「待機」のパッケージ103がなかった場合には、運転している他の計算機101上で停止したパッケージ103と同種類のパッケージを起動してもよい。

【0051】モードが「待機」のパッケージ103があった場合には、ステップ2506において、モード管理機構2302は、モード制御機構2304へ「運転」を指示するモード制御要求(D2304)を送る。そして、モード制御機構2304がクラスタデーモン102を介し、ステップ2505で見つけたパッケージ103へモード制御要求(D2304)を送信することにより、待機していたパッケージ103を運転させる。

【0052】次に、ステップ2510において、モード管理機構2302は管理テーブル2303を更新する。すなわち、ステップ2506で運転を指示したパッケージ103についての管理テーブル2303のモードを「運転」に書き換える。ステップ2510が終了すると、ステップ2502に戻る。

【0053】一方、ステップ2503で通知の種類が計算機101の起動であると判断された場合には、ステップ2507において、モード管理機構2302はモード制御機構2304にモード制御要求(D2304)を送り、起動した計算機101上において停止していたパッケージ103を待機の状態に起動する。次に、ステップ2508において、管理テーブル2303を参照し、モードが「運転」のパッケージ103があるかどうかを調

べる。ここで、複数種類のパッケージ103が存在する場合には、ステップ2507で起動したパッケージ103と同種類のパッケージ103であってモードが「運転」であるパッケージ103があるかどうかを調べる。

【0054】モードが「運転」であるパッケージ103がある場合には、ステップ2510において、モード管理機構2302は管理テーブル2303を更新する。すなわち、ステップ2507で起動したパッケージ103についての管理テーブル2303のモードを「待機」に書き換える。ステップ2510が終了すると、ステップ2502に戻る。

【0055】一方、モードが「運転」であるパッケージ103がない場合には、ステップ2509において、モード管理機構2302は、モード制御機構2304へ「運転」を指示するモード制御要求(D2304)を送る。そして、モード制御機構2304がクラスタデーモン102を介し、ステップ2508で見つけたパッケージ103にモード制御要求(D2305)を送信することにより、待機していたパッケージ103を運転させる。次に、ステップ2510において、モード管理機構2302は管理テーブル2303を更新する。すなわち、ステップ2509で運転を指示したパッケージ103についての管理テーブル2303のモードを「運転」に書き換える。ステップ2510が終了すると、ステップ2502に戻る。

【0056】図26はこの実施の形態の動作の一例を説明する図である。図26において、図1と同一の符号は同一または相当の部分を表している。まず、最初に図26(a)に示すように、第1の計算機である計算機A101a上で第1のパッケージであるパッケージA103aが実行されており、第2の計算機である計算機B101b上で第2のパッケージであるパッケージB103bが待機している状態で、図26(b)に示すように計算機A101aが停止したとすると、上述のようにマネージャ105が動作し、図26(c)に示すように、ただちに計算機B101b上で待機していたパッケージB103bが実行、すなわち運転状態となる。このときパッケージA103aからパッケージB103bへの切り替わりは、パッケージB103bが待機状態で待っていたため、起動にかかる時間が省かれ、高速に行うことができる。そして、計算機A101aが停止状態から復帰し正常に動作し始めると、計算機A101a上では先ほど停止したパッケージA103aが起動し、待機状態となる。ここで、計算機A101aが障害から復帰した後、ただちにパッケージB103bからパッケージA103aに処理を切り替えるのではなく、パッケージA103aを待機状態とするため、新たに切り替え処理が発生せず、システム全体として処理が高速に実行される。

【0057】以上のように、この実施の形態の構成によれば、クラスタシステム上で運転/待機といったモード

により管理される2重系のシステムを構築することができ、パッケージの再起動を行うよりも短い時間で切り替えを行うことができる。

【0058】実施の形態10. 本発明の第10の実施の形態について、図27～図30に基づいて説明する。本実施形態は、図1に示したクラスタシステムのマネージャ105の代わりに用いられ、各パッケージ103の出力を管理する機能を有するマネージャを備えたものである。本実施形態におけるマネージャの構成を図27に示す。図27において、図23と同一の符号は同一又は相当の部分を表す。出力管理機構2701は、通知処理機構2301からのリソース状態変化通知(D2302)を受け、管理テーブル2702を参照/更新し、出力抑止機構(2703)に出力抑止/解除要求を送る(D2702)機構である。出力抑止機構2703は、出力管理機構2701からの出力抑止/解除要求(D2702)を受信し、クラスタデーモン102またはパッケージ103に対して出力抑止/解除要求を送信する(D2703)機構である。

【0059】図28は管理テーブル2702の記憶内容の例を示す。管理テーブル2702には、パッケージ名と出力抑止の状態、例えば抑止、解除が保持されている。図29は、図27のマネージャ105の処理を説明するフローチャートである。

【0060】次に、図29に基づいて、この実施の形態のマネージャ105の動作について説明する。ステップ2901において、出力管理機構2701は管理テーブル2702の初期化を行う。次に、ステップ2902において、通知処理機構2301はクラスタデーモン102からのリソース状態変化通知(D2301)を待ち、受信したら出力抑止機構2703へリソース状態変化通知(D2301)を送信する。続いて、ステップ2903において、出力管理機構2701はリソース状態変化通知(D2301)の種類を判別する。リソース状態変化通知(D2301)が計算機101の起動である場合にはステップ2907へ進み、計算機101の停止である場合にはステップ2904へ進む。計算機の停止であった場合、ステップ2904において、出力管理機構2701は、管理テーブル2702を更新し、停止した計算機105上のすべてのパッケージ103にかかる出力抑止状態を「抑止」に書き換える。

【0061】次に、ステップ2905において、出力管理機構2701は、出力抑止状態が「抑止」のパッケージ103があるかを調べる。このとき、複数種類のパッケージ103が計算機上で実行されている場合には、抑止したパッケージ103と同種類のパッケージ103であって出力抑止状態が「抑止」であるパッケージ103があるかを調べる。そのために、図28に示した管理テーブル2702に、パッケージ103が実行されている計算機の情報及びパッケージ103の種類の情報を追加

してもよい。ステップ2905で、出力抑止状態が「抑止」のパッケージ103がなかった場合にはステップ2902に戻る。ここで、出力抑止状態が「抑止」のパッケージ103がなかった場合には、運転している他の計算機101上で、ステップ2904において抑止されたパッケージ103と同種類のパッケージを起動してもよい。

【0062】出力抑止状態が「抑止」のパッケージ103があった場合には、ステップ2906において、出力管理機構2701は、出力抑止機構2703へ「解除」を指示する出力抑止解除要求(D2702)を送る。そして、出力抑止機構2703がクラスタデーモン102を介し、ステップ2905で見つけたパッケージ103へ出力抑止解除要求(D2703)を送信することにより、出力が抑止されていたパッケージ103が出力を開始する。

【0063】次に、ステップ2910において、出力管理機構2701は管理テーブル2702を更新する。すなわち、ステップ2906で出力抑止解除を指示したパッケージ103についての管理テーブル2702の出力抑止状態を「解除」に書き換える。ステップ2910が終了すると、ステップ2902に戻る。

【0064】一方、ステップ2903で通知の種類が計算機101の起動であると判断された場合には、ステップ2907において、出力管理機構2701は出力抑止機構2703に出力抑止解除要求(D2702)を送り、起動した計算機101上において、パッケージ103を出力が抑止された状態で起動する。次に、ステップ2908において、管理テーブル2702を参照し、出力抑止状態が「解除」のパッケージ103があるかどうかを調べる。ここで、複数種類のパッケージ103が存在する場合には、ステップ2907で起動したパッケージ103と同種類のパッケージ103であって出力抑止状態が「解除」であるパッケージ103があるかどうかを調べる。

【0065】出力抑止状態が「解除」であるパッケージ103がある場合には、ステップ2910において、出力管理機構2701は管理テーブル2702を更新する。すなわち、ステップ2907で起動したパッケージ103についての管理テーブル2702の出力抑止状態を「抑止」に書き換える。ステップ2910が終了すると、ステップ2902に戻る。

【0066】一方、出力抑止状態が「解除」であるパッケージ103がない場合には、ステップ2909において、出力管理機構2701は、出力抑止機構2703へ「解除」を指示する出力抑止解除要求(D2702)を送る。そして、出力抑止機構2703がクラスタデーモン102を介し、ステップ2908で見つけたパッケージ103に出力抑止解除要求(D2702)を送信することにより、出力が抑止されていたパッケージ103が

出力を開始する。次に、ステップ2910において、出力管理機構2701は管理テーブル2702を更新する。すなわち、ステップ2909で出力抑止解除を指示したパッケージ103についての管理テーブル2702の出力抑止状態を「解除」に書き換える。ステップ2910が終了すると、ステップ2902に戻る。

【0067】図30はこの実施の形態の動作の一例を説明する図である。図30において、図1と同一の符号は同一または相当の部分を表している。まず、最初に図30(a)に示すように、第1の計算機である計算機A101a上で第1のパッケージであるパッケージA103aが実行されており、第2の計算機である計算機B101b上で第2のパッケージであるパッケージB103bが実行されており、パッケージA103aは出力を行っている、すなわち出力抑止解除状態であり、パッケージB103bは出力が抑止され、すなわち出力抑止状態である。但し、パッケージA103aとパッケージB103bは、同じ入力データを受け取り、同様に動作している。図30(a)に示す状態で、図30(b)に示すように計算機A101aが停止したとすると、上述のようにマネージャ105が動作し、図30(c)に示すように、ただちに計算機B101b上で出力を抑止していたパッケージB103bが出力を開始する。このとき、パッケージB103bはパッケージA103aと同様に動作しているため、切り替えは一瞬で完了し、実施の形態9に示したシステムよりもシステムの回復時間が速いという特徴がある。そして、計算機A101aが停止状態から復帰し正常に動作し始めると、計算機A101a上では先ほどダウンしたパッケージA103aが起動し、再び動作し始める。このとき、出力は抑止された状態で動作する。ここで、計算機A101aが障害から復帰した後も、ただちにパッケージB103bからパッケージA103aに処理を切り替えるのではなく、パッケージA103aを出力抑止状態とするため、新たに切り替え処理が発生せず、システム全体として処理が高速に実行される。

【0068】ここで、出力制御手段は、出力抑止、出力抑止解除を行う手段であり、パッケージプログラムに設けてもよいし、パッケージプログラムとは別のプロセス、又はスレッドで実行されるプログラムであってもよい。パッケージ103が別のパッケージ103等の他のプロセス、スレッド、若しくは周辺機器等のハードウェアコントローラと通信する場合には、出力制御手段を介して通信を行うようにする。

【0069】以上のように、この実施の形態の構成によれば、クラスタシステム上でFree Run Dualシステムを構築することができ、パッケージの再起動を行うよりも短い時間で切り替えを行うことができる。

【0070】実施の形態11. 本発明の第11の実施の形態について、図31～図34に基づいて説明する。本

実施形態は、図1に示したクラスタシステムのマネージャ105の代わりに用いられ、各パッケージをモードで管理する機能を有するマネージャを備えたものである。本実施形態におけるマネージャの構成を図31に示す。図31において、パッケージ管理機構3101は、通知処理機構2301からのリソース状態変化通知(D2302)を受け、管理テーブル3102を参照/更新し、パッケージ制御機構3103にパッケージ制御要求(D3102)を送信する機構である。パッケージ制御機構3103は、パッケージ管理機構3101からのパッケージ制御要求(D3102)を受信し、クラスタデーモン102またはパッケージ103に対してパッケージ制御要求(D3103)を送信する機構である。

【0071】図32は管理テーブル3102の記憶内容の例を示す。管理テーブル3102には、図32(a)に示すように、パッケージ名、モードの状態、実行計算機、グループ名が記憶されている第1のテーブルと、図32(b)に示すように各計算機の動作状態、例えば「運転」、「停止」の情報が記憶されている第2のテーブルが保持されている。ここで、モードの状態の情報は例えば「運転」、「停止」、「待機」等の情報であり、実行計算機の情報はパッケージ名で特定されるパッケージ103が、どの計算機101で実行されているかを示す情報、グループ名は、パッケージの種類を示す情報であり、同一のグループ名が指定されたパッケージ同士は、他のパッケージの処理を引き継ぐことができる。図33は、図31のマネージャ105の処理を説明するフローチャートである。

【0072】次に、図33に基づいて、マネージャ105の動作について説明する。ステップ3201において、パッケージ管理機構3101は管理テーブル3102の初期化を行う。次に、ステップ3202において、通知処理機構2301はクラスタデーモン102からのリソース状態変化通知(D2301)を待ち、受信したらパッケージ管理機構3101へリソース状態変化通知(D2302)を送信する。続いて、ステップ3203において、パッケージ管理機構3101はリソース状態変化通知(D2302)の種類を判別する。リソース状態変化通知(D2302)が計算機101の起動である場合にはステップ3208へ進み、計算機101の停止である場合にはステップ3204へ進む。計算機の停止であった場合、ステップ3204において、パッケージ管理機構3101は、管理テーブル3102を更新し、停止した計算機105上で運転または待機していたすべてのパッケージ103のモードを「停止」に書き換える。

【0073】次に、ステップ3205において、停止したパッケージのグループと同一のグループ内でモードが「待機」のパッケージ103があるかを調べる。ステップ3205で、モードが「待機」のパッケージ103が

なかった場合にはステップ 3208 に跳ぶ。

【0074】モードが「待機」のパッケージ 103 があった場合には、ステップ 3206 において、パッケージ管理機構 3101 は、パッケージ制御機構 3103 へ「運転」を指示するパッケージ制御要求 (D3102) を送る。そして、パッケージ制御機構 3103 がクラスターデーモン 102 を介し、ステップ 3205 で見つけたパッケージ 103 へパッケージ制御要求 (D3102) を送信することにより、待機していたパッケージ 103 を運転させる。

【0075】次に、ステップ 3207 において、パッケージ管理機構 3101 は管理テーブル 3102 を更新する。すなわち、ステップ 3206 で運転を指示したパッケージ 103 についての管理テーブル 3102 のモードを「運転」に書き換える。ステップ 3210 が終了すると、ステップ 3202 に戻る。

【0076】続いて、ステップ 3208 において、パッケージ管理機構 3101 は管理テーブル 3102 を参照し、モードが「停止」であるパッケージを実行することができる計算機 101 があるか否かを判断する。計算機 101 がある場合には次のステップ 3209 に移り、ない場合にはステップ 3212 へ跳ぶ。

【0077】ステップ 3209 において、パッケージ管理機構 3101 はパッケージ制御機構 3103 へ「待機」を指示するパッケージ制御要求 (D3102) を送る。そして、パッケージ制御機構 3103 がステップ 3208 で見つけた計算機 101 のクラスターデーモン 102 へパッケージ制御要求 (D3103) を送信する。クラスターデーモン 102 は、「待機」を指示するパッケージ制御要求 (D3103) を受け取ると、目的のパッケージ 103 を待機状態で起動させる。次に、ステップ 3210 において、パッケージ管理機構 3101 は管理テーブル 3102 を参照し、ステップ 3209 で起動したパッケージのグループと同一のグループでモードが「運転」であるパッケージがあるか否かを調べ、ある場合にはステップ 3212 へ跳び、ない場合には、次のステップ 3211 へ移る。

【0078】次にステップ 3211 において、パッケージ管理機構 3101 はパッケージ制御機構 3103 へ「運転」を指示するパッケージ制御要求 (D3102) を送る。そして、パッケージ制御機構 3103 が、ステップ 3209 で起動したパッケージ 103 を管理するクラスターデーモン 102 へパッケージ制御要求 (D3103) を送信する。クラスターデーモン 102 は、「運転」を指示するパッケージ制御要求 (D3103) を受け取ると、目的のパッケージ 103 の待機状態を解除し、運転を開始させる。

【0079】次に、ステップ 3212 において、パッケージ管理機構 3101 は管理テーブル 3102 を更新する。すなわち、ステップ 3209、ステップ 3211 で

変更されたパッケージ 103 の状態を管理テーブル 3102 に反映させる。ステップ 3212 が終了すると、ステップ 3202 に戻る。

【0080】図 34 はこの実施の形態の動作の一例を説明する図である。図 34 において、図 1 と同一の符号は同一または相当の部分を表している。まず、最初に図 34 (a) に示すように、第 1 の計算機である計算機 A 101 a 上で第 1 のパッケージであるパッケージ A 103 a が運転されており、第 2 の計算機である計算機 B 101 b 上で第 2 のパッケージであるパッケージ B 103 b が待機しており、パッケージ A 103 a とパッケージ B 103 b は同一のグループであるとする。図 34 (a) に示す状態で、図 34 (b) に示すように計算機 A 101 a が停止したとすると、上述のようにマネージャ 105 が動作し、図 34 (c) に示すように、ただちに計算機 B 101 b 上で待機していたパッケージ B 103 b が実行、すなわち運転状態となり、パッケージ A 103 a の処理を引き継ぐ。このときパッケージ A 103 a からパッケージ B 103 b への切り替わりは、パッケージ B 103 b が待機状態で待っていたため、起動にかかる時間が省かれ、高速に行うことができる。そして、第 3 の計算機である別の計算機 C 101 c 上では先ほど停止したパッケージ A 103 a と同一のパッケージ 103 が起動し、待機状態となる。

【0081】以上のように、この実施の形態の構成によれば、クラスタシステム上で運転/待機といったモードにより管理される 2 重系のシステムを構築することができ、パッケージの再起動を行うよりも短い時間で切り替えを行うことができる。さらに、この実施の形態の構成によれば、縮退運転時間を短くすることができるという効果がある。すなわち、従来の 2 重系のシステムでは、片方の計算機が停止した場合、その計算機の復旧作業を行っている間は、1 重系となり、その間にもう一方の計算機が停止するとシステムダウンとなったが、この実施の形態によれば、一方のパッケージが停止した場合、停止したパッケージは他の計算機で起動されるため、1 重系となる時間はパッケージの再起動時間のみとなり、多重障害によるシステムダウンを避けることができ、システムの信頼性を高めることができる。

【0082】実施の形態 12. 本発明の第 12 の実施の形態について、図 35 ~ 図 37 に基づいて説明する。本実施形態のマネージャ 105 の構成は、図 31 に示したものと基本的に同様であるが、以下に説明するように各構成で実行される処理が異なる。

【0083】まず、管理テーブル 3102 について説明する。図 35 は管理テーブル 3102 の記憶内容の例を示す。この実施の形態の管理テーブル 3102 には、図 35 (a) に示すように、パッケージ名、そのパッケージが実行されている実行計算機名が記憶された第 3 のテーブルと、図 35 (b) に示すように各計算機の動作状態、

例えば「運転」、「停止」の情報が記憶されている第2のテーブルが保持されている。

【0084】次に、図33に基づいて、マネージャ105の動作について説明する。図33は、図31のマネージャ105の処理を説明するフローチャートである。ステップ3501において、パッケージ管理機構3101は管理テーブル3102の初期化を行う。次に、ステップ3502において、通知処理機構2301はクラスタデーモン102からのリソース状態変化通知(D2301)を待ち、受信したらパッケージ管理機構3101へリソース状態変化通知(D2302)を送信する。続いて、ステップ3503において、パッケージ管理機構3101はリソース状態変化通知(D2302)の種類を判断する。リソース状態変化通知(D2302)が計算機101の起動である場合にはステップ3507へ進み、計算機101の停止である場合にはステップ3504へ進む。計算機の停止であった場合、ステップ3504において、パッケージ管理機構3101は、管理テーブル3102を参照し、空き計算機101を検索する。空き計算機101が見つかった場合には、次のステップ3506へ進み、空き計算機101がなかった場合には、ステップ3510へ跳ぶ。

【0085】次に、ステップ3506において、パッケージ管理機構3101は、パッケージ制御機構3103へ「運転」を指示するパッケージ制御要求(D3102)を送る。そして、パッケージ制御機構3103がステップ3505で見つけた計算機101のクラスタデーモン102へ、「運転」を指示するパッケージ制御要求(D3103)を送る。このパッケージ制御要求(D3103)を受け取ったクラスタデーモン102は、停止した計算機101上で実行されていたパッケージ103と同一のパッケージ103を起動させ、パッケージ103の実行を開始させる。次に、ステップ3510において、パッケージ管理機構3101は管理テーブル3102の内容を現在の計算機の状態に更新し、ステップ3502へ戻る。

【0086】一方、ステップ3503で通知が計算機101の起動であると判断された場合には、ステップ3507において、パッケージ管理機構3101は管理テーブル3102の内容を現在の計算機101の状態に更新、すなわち、起動した計算機101にかかる状態の項目を「運転」に書き換える。次に、ステップ3508において、停止しているパッケージ103があるかを、例えば、実施の形態11で説明した図32(a)に示すような第1のテーブルを参照することによって調べ、停止しているパッケージがある場合には、停止しているパッケージ103を実行できる計算機101があるかを調べる。停止しているパッケージ103がない場合、若しくは停止しているパッケージ103を実行できる計算機103がない場合には、上述のステップ3510へ跳び、

停止しているパッケージ103があり、かつ停止しているパッケージ103を実行できる計算機101がある場合には、次のステップ3509へ進む。

【0087】ステップ3509において、パッケージ管理機構3101は、パッケージ制御機構3103へ「運転」を指示するパッケージ制御要求(D3102)を送る。そして、パッケージ制御機構3103がステップ3508で見つけた計算機101のクラスタデーモン102へ、「運転」を指示するパッケージ制御要求(D3103)を送信することにより、ステップ3508で見つけた計算機101上で、同じくステップ3508で見つけたパッケージ103を起動させるとともに、実行させる。ステップ3509が終了すると、上述のステップ3510へ進む。

【0088】図37はこの実施の形態の動作の一例を説明する図である。図37において、図1と同一の符号は同一または相当の部分を表している。まず、最初に図37(a)に示すように、計算機A101a上でパッケージA103aが運転されており、計算機B101b上でパッケージB103bが運転されており、パッケージA103aとパッケージB103bは異なる種類のパッケージ103であるとする。図34(a)に示す状態で、図34(b)に示すように計算機A101aが停止したとすると、上述のようにマネージャ105が動作し、パッケージA103aが空き計算機N101n上で起動され、実行される。その後、図34(c)に示すように、計算機A101aが起動すると、今度は計算機A101aが空き計算機、すなわちバックアップ用の計算機となる。そして、図37(d)に示すように、今度は計算機B101bが故障等により停止すると、上述のようにマネージャ105が動作し、計算機B101b上で実行されていたパッケージB103bが、今度は空き計算機となっていた計算機A101a上で起動され、実行される。

【0089】以上のように、この実施の形態の構成によれば、クラスタシステム上でシステムの状態によりパッケージの移動先を変更するような多重系システムを構築することができる。また、この実施の形態では、図37(c)のように停止した計算機A101aが回復しても、計算機A101aをバックアップして、パッケージA103aの実行を行った計算機N101nがそのまま実行を続け、回復した計算機A101aは今度は他の計算機のバックアップに回る。そのため、回復した計算機A101aが再びパッケージA103aを実行し、計算機N101nが再びバックアップに回るようなシステムに比べ、計算機の切り替え回数が減り、システム全体の処理パフォーマンスが向上する。すなわち、処理を高速に実行することができる。

【0090】実施の形態13. 本発明の第13の実施の形態について、図38及び図39に基づいて説明する。本実施形態のマネージャ105の構成は、図31に示し

たものと基本的に同様であり、その処理は基本的に図36を用いて説明した実施の形態12と同様である。ただし、以下に説明するようにパッケージ103を起動する計算機101を求める処理、すなわち、ステップ3504及びステップ3508が異なる。

【0091】まず、この実施の形態の管理テーブル3102について説明する。図38に管理テーブル3102の記憶内容の例を示す。この実施の形態の管理テーブル3102には、図38(a)に示すように、パッケージ名、そのパッケージが実行されている実行計算機名、及びそのパッケージのグループ名が記憶された第4のテーブルと、図38(b)に示すように各計算機の動作状態、例えば「運転」、「停止」の情報が記憶されている第2のテーブルが保持されている。

【0092】次に、この実施の形態のマネージャ105の動作について説明する。上述のように、この実施の形態のマネージャ105の動作は、基本的に実施の形態12と同様であるため、異なる処理、すなわち、ステップ3504及びステップ3508について以下に説明する。この実施の形態のステップ3504においては、パッケージ管理機構3101は、管理テーブル3102を参照し、空き計算機ではなく、停止した計算機101上で実行されていたパッケージ103と同一のグループのパッケージ103が実行されていない計算機101を検索する。同一のグループのパッケージ103が実行されていない計算機101が見つかった場合には、次のステップ3506へ進み、見つからなかった場合には、ステップ3510へ跳ぶ。

【0093】また、この実施の形態のステップ3508においては、停止しているパッケージ103があるかを、例えば、実施の形態11で説明した図32(a)に示すような第1のテーブルを参照することによって調べ、停止しているパッケージがある場合には、停止しているパッケージ103を実行できる計算機101があるかを調べる。停止しているパッケージ103がない場合、若しくは停止しているパッケージ103を実行できる計算機103がない場合には、上述のステップ3510へ跳び、停止しているパッケージ103があり、かつ停止しているパッケージ103を実行できる計算機101がある場合には、次のステップ3509へ進む。ここで、実行できる計算機101として、空き計算機101ではなく、停止しているパッケージ103と同一のグループのパッケージ103が実行されていない計算機101を見つける。

【0094】図39はこの実施の形態の動作の一例を説明する図である。図39において、図1と同一の符号は同一または相当の部分を表している。まず、最初に図39(a)に示すように、計算機A101a上でパッケージA103aが運転されており、計算機B101b上でパッケージB103bが運転されており、パッケージA

103aとパッケージB103bは、同一グループに指定されたパッケージであり、一つのまとまった処理をそれぞれのパッケージで分散して行う、すなわち並列処理するパッケージであり、計算機A101aで実行されているパッケージC103cは、パッケージA103aとパッケージB103bの出力をまとめるためのパッケージであるとする。図34(a)に示す状態で、図34(b)に示すように計算機A101aが停止したとすると、上述のようにマネージャ105が動作し、パッケージA103aが同一グループのパッケージB103bが実行されている計算機B101bではなく、同一グループのパッケージが実行されていない計算機N101nで起動され、実行される。

【0095】以上のように、この実施の形態の構成によれば、クラスタシステム上でロードシェアシステムを構築することができ、負荷の分散を行うことができる。すなわち、並行処理を行っている複数のパッケージ同士を同一の計算機101上で起動しないことにより、切り替え処理が実行された後も並列処理による高速処理を継続することができる。

【0096】実施の形態14. 本発明の第14の実施の形態について、図40～図42に基づいて説明する。本実施形態のマネージャ105の構成は、図31に示したものと基本的に同様であるが、以下に説明するように各構成で実行される処理が異なる。

【0097】まず、管理テーブル3102について説明する。図40は管理テーブル3102の記憶内容の例を示す。この実施の形態の管理テーブル3102には、図40(a)に示すように、パッケージ名、そのパッケージが実行されている実行計算機名、及びそのパッケージの優先順位が記憶された第5のテーブルと、図40(b)に示すように各計算機の動作状態、例えば「運転」、「停止」の情報が記憶されている第2のテーブルが保持されている。

【0098】次に、図41に基づいて、マネージャ105の動作について説明する。図41は、図31のマネージャ105の処理を説明するフローチャートである。ステップ4001において、パッケージ管理機構3101は管理テーブル3102の初期化を行う。次に、ステップ4002において、通知処理機構2301はクラスタデーモン102からのリソース状態変化通知(D2301)を待ち、受信したらパッケージ管理機構3101へリソース状態変化通知(D2302)を送信する。続いて、ステップ4003において、パッケージ管理機構3101はリソース状態変化通知(D2302)の種類を判断する。リソース状態変化通知(D2302)が計算機101の起動である場合にはステップ4004へ進み、計算機101の停止である場合にはステップ4007へ進む。

【0099】一方、ステップ4003で通知が計算機1

01の起動であると判断された場合には、ステップ4004において、パッケージ管理機構3101は管理テーブル3102を参照し、停止しているパッケージ103を検索する。ここで、停止しているパッケージ103を検索する方法の一例としては、図40(a)の第5のテーブルの実行計算機で指定された計算機が、図40(b)の第2のテーブルにおいて「停止」となっているパッケージ103、若しくは、図40(a)の第5のテーブルにパッケージ名及び優先順位は指定されているものの、実行計算機の項目にはどの計算機も指定されていないパッケージ103を検索する方法がある。

【0100】次に、ステップ4005において、パッケージ管理機構3101はステップ4004で停止しているパッケージが見つかったか否かを判断する。ここで、停止しているパッケージがあった場合には、次のステップ4006へ進み、ない場合にはステップ4012へ跳ぶ。

【0101】次に、ステップ4006において、パッケージ管理機構3101は、まず、停止しているパッケージ103のうち一番優先順位の高いパッケージを探し、パッケージ制御機構3103へ一番優先順位の高いパッケージ103の「運転」を指示するパッケージ制御要求(D3102)を送る。そして、パッケージ制御機構3103が、新たに起動した計算機101のクラスタデーモン102へ、「運転」を指示するパッケージ制御要求(D3103)を送信することにより、新たに起動した計算機101上で、停止しているパッケージ103のうち一番優先順位の高いパッケージ103が実行される。ステップ4006が終了すると、次のステップ4012へ移る。

【0102】次に、ステップ4012において、パッケージ管理機構3101は管理テーブル3102の内容を、現在の計算機及びパッケージの状態に合わせて更新し、ステップ4002へ戻る。

【0103】一方、ステップ4003で計算機の停止であると判断された場合、ステップ4007において、パッケージ管理機構3101は、管理テーブル3102を参照し、停止した計算機101上で動作していたパッケージを実行可能な計算機101を検索する。実行可能な計算機101が見つかった場合には、次のステップ4011へ進み、実行可能な計算機101がなかった場合には、ステップ4009へ跳ぶ。ここで、パッケージ103を実行できるか否かは、そのパッケージの実行に必要なリソースと、対象計算機101の残りリソースを比較することにより行い、計算機101にパッケージの実行に必要なリソースが十分に残っている場合には実行可能であるとし、残っていない場合には実行不能と判断する。

【0104】実行可能なパッケージ103がない場合には、ステップ4009において、パッケージ管理機構3

101は管理テーブル3102を参照し、停止したパッケージ103よりも優先順位の低いパッケージ103で停止可能なパッケージ103があるか否かを調べる。停止可能なパッケージがあった場合には、ステップ4010へ進み、ない場合には上述のステップ4012へ進む。停止可能なパッケージがある場合には、ステップ4010において、パッケージ管理機構3101は、パッケージ制御機構3103へ「停止」を指示するパッケージ制御要求(D3102)を送る。そして、パッケージ制御機構3103がステップ4009で見つけたパッケージ103を管理するクラスタデーモン102へ、「停止」を指示するパッケージ制御要求(D3103)を送る。このパッケージ制御要求(D3103)を受け取ったクラスタデーモン102は、ステップ4009で見つかったパッケージ103を停止させ、そのパッケージが使用していたリソースを開放させる。このステップ4010が終了すると、ステップ4011へ移る。

【0105】ステップ4011においては、パッケージ管理機構3101は、パッケージ制御機構3103へ、停止したパッケージ103の「運転」を指示するパッケージ制御要求(D3102)を送る。そして、パッケージ制御機構3103が、ステップ4008で見つけた計算機101、若しくはステップ4010でパッケージ103を停止させた計算機101のクラスタデーモン102へ、「運転」を指示するパッケージ制御要求(D3103)を送る。このパッケージ制御要求(D3103)を受け取ったクラスタデーモン102は、停止したパッケージ103を起動し、運転を開始させる。ステップ4011が終了すると、上述のステップ4012へ移る。

【0106】図42はこの実施の形態の動作の一例を説明する図である。図42において、図1と同一の符号は同一または相当の部分を表している。まず、最初に図42(a)に示すように、計算機A101a～計算機N101nのそれぞれでパッケージA103a～パッケージN101nが運転されており、パッケージA103a～パッケージN101nはそれぞれ異なる種類のパッケージ103であるとする。各パッケージの優先順位は、パッケージA103aが最も高く、その次にパッケージB103b、パッケージN101nが最も低いとする。図42(a)に示す状態で、図34(b)に示すように第1の計算機である計算機A101aが停止したとすると、上述のようにマネージャ105が動作し、計算機N101n上で、パッケージA103aよりも優先順位の低いパッケージN101nが停止され、代わりに停止したパッケージA103aが起動され、実行される。

【0107】その後、図34(c)に示すように、計算機A101aが回復し、起動すると、今度は計算機A101a上で、停止していたパッケージN101nが実行される。そして、今度は図37(d)に示すように、計算機B101bが故障等により停止すると、上述のように

マネージャ 105 が動作し、計算機 A 101 a で実行されていた優先順位の低いパッケージ N 101 n が停止し、その代わりに優先順位の高い停止したパッケージ B 103 b が起動され、実行される。

【0108】ここで、さらに図 4 2 (e) に示すように、計算機 N 101 n までもが停止すると、パッケージ A 103 a はパッケージ B 103 b よりも優先順位が高いため、計算機 A 101 a 上で実行されていたパッケージ B 103 b が停止し、代わりにパッケージ A 103 a が計算機 A 101 a 上で起動され、実行される。その後、図 4 2 (f) に示すように、計算機 B 101 b が回復すると、停止しているパッケージの中で一番優先順位の高いパッケージ B 103 b が、計算機 B 101 b 上で起動され、実行される。

【0109】以上のように、この実施の形態の構成によれば、クラスタシステム上で多重縮退システムを構築することができ、障害発生時にも重要な処理を優先的に行うことができる。

【0110】実施の形態 15. 本発明の第 15 の実施の形態について、図 4 3 ~ 図 4 6 に基づいて説明する。本実施形態のマネージャ 105 の構成は、図 3 1 に示したものと基本的に同様であるが、以下に説明するように各構成で実行される処理が異なる。

【0111】まず、管理テーブル 3 102 について説明する。図 4 3 は管理テーブル 3 102 の記憶内容の例を示す。この実施の形態の管理テーブル 3 102 には、図 4 3 (a) に示すように、パッケージ名、そのパッケージが実行されている実行計算機名、のパッケージの優先順位、及びそのパッケージの実行に必要な負荷が記憶された第 6 のテーブルと、図 4 3 (b) に示すように各計算機の動作状態、例えば「運転」、「停止」の情報、及び各計算機の最大許容負荷が記憶されている第 7 のテーブルが保持されている。負荷の例としてはリソース、例えば、メモリ、CPU 時間等がある。

【0112】次に、図 4 4 に基づいて、マネージャ 105 の動作について説明する。図 4 1 は、図 3 1 のマネージャ 105 の処理を説明するフローチャートである。ステップ 4 3 0 1 において、パッケージ管理機構 3 101 は管理テーブル 3 102 の初期化を行う。次に、ステップ 4 3 0 2 において、通知処理機構 2 3 0 1 はクラスタデーモン 102 からのリソース状態変化通知 (D 2 3 0 1) を待ち、受信したらパッケージ管理機構 3 101 へリソース状態変化通知 (D 2 3 0 2) を送信する。続いて、ステップ 4 3 0 3 において、パッケージ管理機構 3 101 はリソース状態変化通知 (D 2 3 0 2) の種類を判断する。リソース状態変化通知 (D 2 3 0 2) が計算機 101 の起動若しくは停止である場合にはステップ 4 3 0 9 へ進み、計算機 101 の負荷 100 % 以上を伝える通知である場合には次のステップ 4 3 0 4 へ進む。

【0113】ステップ 4 3 0 4 において、負荷が 100

% 以上となった計算機 101 (以下、この実施の形態において、過負荷計算機という) 上で動作していたパッケージ 103 であって、処理を行っていないパッケージ 103 のうち優先度の低いものを選択し、そのパッケージ 103 (以下、この実施の形態において選択パッケージという) について、以下のステップ 4 3 0 5 ~ 4 3 0 8 の処理を行う。

【0114】まず、ステップ 4 3 0 5 において、選択パッケージを停止する。すなわち、パッケージ管理機構 3 101 は、パッケージ制御機構 3 103 へ選択パッケージの停止を指示するパッケージ制御要求 (D 3 102) を送る。そして、パッケージ制御機構 3 103 が、当該選択パッケージを管理するクラスタデーモン 102 へ「停止」を指示するパッケージ制御要求 (D 3 103) を送信することにより、この要求を受け取ったクラスタデーモン 102 が選択パッケージ停止させる。ここで、選択パッケージを停止させることにより、過負荷計算機のリソースが開放され、過負荷計算機の負荷が減少する。

【0115】次に、ステップ 4 3 0 6 において、パッケージ管理機構 3 101 は、選択パッケージにかかる管理テーブル 2 3 0 3 の項目を更新、すなわち「停止」に書き換える。ステップ 4 3 0 7 において、過負荷計算機上で動作する全パッケージについて、上述のステップ 4 3 0 4 ~ 4 3 0 6 の処理を行ったかを判断し、処理した場合には、ステップ 4 3 10 へ進み、処理していない場合には、次のステップ 4 3 0 8 へ移る。

【0116】次のステップ 4 3 0 8 において、過負荷計算機の負荷が 100 % 以上であるかが調べられる。100 % 以上である場合には、上述ステップ 4 3 0 4 へ戻り、100 % 未満である場合には、ステップ 4 3 10 へ移る。100 % 以上であるか否かは、パッケージ管理機構 3 101 が、管理テーブル 2 3 0 3 を参照し、過負荷計算機で運転或いは待機しているパッケージの負荷の合計と、過負荷計算機の最大許容負荷とを比較することにより判断する。

【0117】一方、ステップ 4 3 0 3 で通知の種類が計算機の起動若しくは停止であると判断された場合には、ステップ 4 3 0 9 において、パッケージ管理機構 3 101 は、管理テーブル 2 3 0 3 を更新し、当該計算機の項目を通知の内容に応じて、「運転」若しくは「停止」に書き換える。次に、ステップ 4 3 10 において、パッケージ管理機構 3 101 は管理テーブル 3 102 を参照し、停止しているパッケージ 103 のうち最も優先順位の高いもの (以下、この実施の形態において、優先パッケージという) を検索する。

【0118】次に、ステップ 4 3 11 において、パッケージ管理機構 3 101 は、管理テーブル 2 3 0 3 を参照し、優先パッケージを実行できる計算機 101 を選択する。ここで、計算機 101 の選択は、優先パッケージの

実行に必要な負荷、各計算機101の最大許容負荷、並びに、各計算機101において運転又は待機しているパッケージにかかる負荷を基準にして行われる。実行できる計算機101がない場合には選択は行われず、実行できる計算機がないという結果が得られる。

【0119】続いて、ステップ4312において、ステップ4311で実行可能な計算機101があったか否かが判断され、あった場合には次のステップ4313に移り、ない場合にはステップ4316へ移る。次のステップ4313においては、ステップ4311で選択された計算機101上で、優先パッケージが起動される。すなわち、パッケージ管理機構3101は、パッケージ制御機構3103へ優先パッケージの「運転」を指示するパッケージ制御要求(D3102)を送る。そして、パッケージ制御機構3103が、ステップ4311で選択された計算機101のクラスタデーモン102へ、「運転」を指示するパッケージ制御要求(D3103)を送信することにより、当該計算機101上で、停止しているパッケージ103のうち一番優先順位の高いパッケージ103が実行される。

【0120】次に、ステップ4314において、パッケージ管理機構3101は管理テーブル2303を更新し、管理テーブル2303の優先パッケージにかかる項目、すなわち実行計算機を更新する。次に、ステップ4315において、パッケージ管理機構3101は停止しているパッケージ103すべてについて、上述4310からの処理、すなわちステップ4310～4318の処理を行ったかを判断し、すべてのパッケージ103について処理した場合には、ステップ4302へ移り、新たなリソース状態変化通知(D2301)が来るのを待つ、すべてのパッケージ103について処理が終わっていない場合には、ステップ4310に戻る。

【0121】一方、ステップ4312で実行可能な計算機101がないと判断された場合には、ステップ4316において、パッケージ管理機構3101は管理テーブル3102を参照し、停止しているパッケージ103よりも優先順位の低いパッケージ103で停止可能なパッケージ103があるか否かを調べる。停止可能なパッケージがあった場合には、ステップ4317へ進み、ない場合には上述のステップ4315へ進む。停止可能なパッケージがある場合には、ステップ4317において、パッケージ管理機構3101は、パッケージ制御機構3103へ「停止」を指示するパッケージ制御要求(D3102)を送る。そして、パッケージ制御機構3103がステップ4316で見つけたパッケージ103を管理するクラスタデーモン102へ、「停止」を指示するパッケージ制御要求(D3103)を受け取ったクラスタデーモン102は、ステップ4316で見つかったパッケージ103を停止させ、そのパッケージが使用していたリソ

ースを開放させる。次に、ステップ4318において、パッケージ管理機構3101は、管理テーブル2303を更新し、ステップ4317で停止させたパッケージにかかる項目を現在の状態に書き換える。ステップ4311が終了すると、上述のステップ4311へ移り、上述のような新たなリソースの開放、或いは優先パッケージの起動の処理を行う。

【0122】図45はこの実施の形態の動作の一例を説明する図である。図45において、図1と同一の符号は同一または相当の部分を表している。まず、最初に図45(a)に示すように、計算機A101a～計算機N101nのそれぞれでパッケージA103a～パッケージF101fが運転されており、パッケージA103a～パッケージF101fはそれぞれ異なる種類のパッケージ103であるとする。各パッケージの優先順位は、パッケージA103aが最も高く、その次にパッケージB103b、以降順に、パッケージC101c、パッケージD101d、パッケージE101e、パッケージF101fであるとする。また、パッケージA103aは計算機A101aの負荷の40%を占めており、パッケージB103b～パッケージF101fはそれぞれ順番に、実行される計算機101の負荷の40%、40%、20%、20%、20%を占めている。図45(a)に示す状態で、図45(b)に示すように第1の計算機である計算機A101aの負荷が100%以上となったとすると、上述のようにマネージャ105が動作し、図45(c)に示すように、計算機A101aで一番優先順位の低い優先順位の低いパッケージD103dが停止し、代わりに、負荷に余裕のある計算機B101b上でパッケージD103dが起動し、実行を開始する。

【0123】また、図46は、この実施の形態の動作の他の例を説明する図である。図46において、図45と同一の符号は同一または相当の部分を表している。図46(a)に示す状態で、図46(b)に示すように計算機A101aが停止して、パッケージA103a及びパッケージD103dが停止したとする。すると、上述のようにマネージャ105が動作し、図46(c)に示すように、停止したパッケージで優先順位の最も高いパッケージA103aが計算機N101n上で起動し実行を開始する。このとき、優先順位の低いパッケージF103fは停止する。一方、停止したパッケージの中で次に優先順位の高いD101dは、負荷に余裕のある計算機B101bで起動され、実行される。また、パッケージF101fは負荷に余裕のある計算機101がなく、優先順位も起動中の他のパッケージ103よりも低いため、起動されない。

【0124】以上のように、この実施の形態の構成によれば、自動負荷分散システムを構築することができ、負荷を自動的に分散し、重要な処理のレスポンスの劣化を防ぐことができる。

【0125】実施の形態16. 本発明の第16の実施の形態について、図47～図49に基づいて説明する。本実施形態は、図1に示したクラスタシステムのマネージャ105の代わりに用いられ、各パッケージに割り当てるリソースの量を、各パッケージの優先順位に基づいて管理する機能を有するマネージャを備えたものである。本実施形態におけるマネージャの構成を図47に示す。図47において、図23と同一の符号は同一又は相当の部分を表す。リソース割り当て管理機構4601は、通知処理機構2301からのリソース状態変化通知(D2301)を受け、管理テーブル4602を参照/更新し、リソース割り当て制御機構4603にパッケージ制御要求(D4603)を送信する機構である。リソース割り当て制御機構4603は、リソース割り当て管理機構4601からのパッケージ制御要求(D4602)を受信し、クラスタデーモン102またはパッケージ103に対してパッケージ制御要求(D4603)を送信することにより、残りのリソースをパッケージの優先順位に応じた量だけ、そのパッケージ103に割り当てる機構である。

【0126】図48は管理テーブル4602の記憶内容の例を示す。管理テーブル4602には、図48に示すように、パッケージ名、実行計算機、優先順位が記憶されている第8のテーブルが保持されている。ここで、実行計算機の情報にはパッケージ名で特定されるパッケージ103が、どの計算機101で実行されているかを示す情報である。図49は、図47のマネージャ105の処理を説明するフローチャートである。

【0127】次に、図49に基づいて、マネージャ105の動作について説明する。ステップ4801において、リソース割り当て管理機構4601は管理テーブル4602の初期化を行う。次に、ステップ4802において、通知処理機構2301はクラスタデーモン102からのリソース状態変化通知(D2301)を待ち、受信したらリソース割り当て管理機構4601へリソース状態変化通知(D2302)を送信する。

【0128】続いて、ステップ4803において、リソース割り当て管理機構4601は、リソース状態変化通知(D2302)に応じて、管理テーブル4602を更新する。ステップ4804において、クラスタシステム内の複数の計算機101のうちで、まだステップ4805～ステップ4808の処理を行っていない計算機を1つ選択する(以下、この実施の形態において、この選択された計算機を選択計算機という)。そして、選択計算機について、以下のステップ4804～ステップ4807の処理を行う。

【0129】まず、ステップ4805において、選択計算機上で実行されているパッケージで、まだステップ4806及びステップ4807の処理を行っていないパッケージのうち優先順位の最も高いものを選択する(以

下、この実施の形態において、この選択されたパッケージを選択パッケージという)。そして、選択パッケージについて、以下のステップ4806及びステップ4807の処理を行う。

【0130】ステップ4806において、選択パッケージに割り当てるリソースの量を選択計算機のリソースの残量から計算する。例えば、

$$[\text{割り当てるリソースの量}] = [\text{リソースの残量} \times 0.5]$$

のような式により求める。そして、あらたなリソースの残量を以下のように求める。

$$(\text{新たな})[\text{リソースの残量}] = [\text{リソースの残量}] - [\text{割り当てるリソースの量}]$$

【0131】次に、ステップ2807において、リソース割り当て管理機構4601は、リソース割り当て制御機構4603へリソース割り当て制御要求(D4602)を送り、リソース割り当て制御機構4603が、選択計算機のクラスタデーモン102へリソース割り当て制御要求(D4603)を送る。リソース割り当て制御要求(D4603)を受け取ったクラスタデーモン102は、選択パッケージにステップ4806で計算された[割り当てるリソースの量]のリソースを割り当てて、選択パッケージを起動し、実行させる。

【0132】続いて、ステップ4808において、選択計算機上のすべてのパッケージ103について、上述ステップ4806～ステップ4807の処理を実行したかを判断し、実行していなかった場合には、ステップ4805に戻り、上述の処理を繰り返す。一方、実行していた場合には、ステップ4809において、全計算機について、上述ステップ4804～ステップ4808の処理を実行したかを判断し、実行していない場合には、ステップ4804に戻り上述の処理を繰り返す。実行した場合には、ステップ4802に戻り、次のリソース状態変化通知(D2301)を待つ。

【0133】なお、このステップ4809の処理が終了すると、ステップ4804において、ステップ4805～ステップ4808の処理を行ったという情報はクリアされ、処理を行っていないものとして取り扱われる。そして、ステップ4802で新たにリソース状態変化通知(D2301)を受け取った場合には、受け取る以前にステップ4805～ステップ4807の処理を行った計算機101についても、ステップ4805～ステップ4807の処理が実行される。ステップ4805のパッケージ103の選択についても同様である。このようにして、クラスタシステムの最新のリソース状態に応じて、パッケージ103にリソースを動的に割り当てる。

【0134】なお、ステップ4806の割り当てるリソース量の計算式は、パッケージ103の優先順位に応じて、優先順位が高いほど多くのリソースが割り当てられるような式であれば、他の式を用いてもよい。例えば、

10

20

30

40

50

〔割り当てるリソースの量〕＝〔リソースの残量〕×（1／〔優先順位〕）×0.8

のような式を用いてもよい。

【0135】以上の構成により、クラスタシステム内で、重要な処理のレスポンスの劣化を防ぐことができ、あまり重要でない処理の継続もすることができる。

【0136】

〔発明の効果〕本発明は、以上説明したように構成されているので、以下に示すような効果を奏する。

【0137】マネージャがクラスタ上の全リソースの監視制御を一括して行ない、パッケージプログラムはマネージャに対してのみアクセスすれば良いようにしたので、障害発生時においても計算機上の動作環境を意識する必要がない。

【0138】また、マネージャを1個のパッケージプログラムとして作成するようにしたので、マネージャあるいはマネージャが動作している計算機上に障害が発生しても、容易に他の計算機上で代替動作させることができる。

【0139】また、各計算機上にその計算機上のリソースを監視制御するエージェントをおくようにしたので、マネージャやネットワークの負荷を軽減することができる。

【0140】また、各計算機上のエージェントが互いに通信し、グローバルデータを直接アクセスするようにしたので、マネージャの搭載が不要となる。

【0141】また、マネージャは同一計算機上のエージェントと通信し、各計算機上のエージェントが互いに通信するようにしたので、パッケージプログラムはマネージャ、およびエージェントのいずれに対してアクセスするかを選択することができる。

【0142】また、自動制御機構を設け、パッケージプログラム間の相互関係を定義した設定ファイルによる実行環境を反映したシステム運転を行うようにしたので、柔軟なシステム設計が可能となる。

【0143】また、パッケージプログラムに動作モードを付加して管理するようにしたので、多重系システムからの移行、およびシステム運用が容易となる。

【0144】また、クラスタシステム内の全リソースの状態の変化を一括してログとして収集保存するようにしたので、障害発生時の解析作業が容易となる。

【0145】また、クラスタ制御システムを構成する複数の計算機のうちの1つの計算機に障害が発生した場合に、上記障害が発生した計算機で運転中のアプリケーションや各種のサービスを提供するパッケージプログラムを他の計算機で運転させるクラスタ制御システムにおいて、上記複数の計算機はそれぞれ、自己の計算機の障害及び回復を監視するとともに、上記パッケージプログラムの起動及び運転を制御するクラスタデーモンを備え、上記複数の計算機のうちの第1の計算機は、上記パッケ

ージプログラムである第1のパッケージプログラムを運転し、上記複数の計算機のうちの第2の計算機は、上記第1のパッケージプログラムと同じアプリケーションやサービスを提供する第2のパッケージプログラムを起動状態で待機させ、上記複数の計算機のうち1つの計算機は、上記クラスタデーモンに加えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記第1の計算機に障害が発生した場合に、上記第2の計算機に上記第2のパッケージプログラムを運転させるとともに、上記第1の計算機が障害から回復した場合には、上記第1の計算機に上記第1のパッケージプログラムを起動状態で待機させるマネージャを備えたため、高速にシステムを回復させることができ、回復後も高速に処理を実行する事ができる。

【0146】また、上記第1のパッケージプログラムの出力若しくは上記第2のパッケージプログラムの出力のいずれかを選択して出力する出力制御手段を有し、上記第2の計算機は、上記第2のパッケージプログラムを起動状態で待機させる代わりに、上記第2のパッケージプログラムを運転し、上記マネージャに代えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記第1のパッケージプログラムの出力が上記出力制御手段から出力されているときに上記第1の計算機で障害が発生した場合、上記第1のパッケージプログラムの出力に代えて上記第2のパッケージプログラムの出力を上記出力制御手段から出力させ、上記第2の計算機で障害が発生するまで上記出力制御手段に上記第2のパッケージプログラムの出力を継続して出力させ、上記第1のパッケージプログラムが運転を再開し上記第2の計算機で障害が発生した場合に、上記第2のパッケージプログラムの出力に代えて、上記第1のパッケージプログラムの出力を上記出力制御手段から出力させるマネージャを備えたため、高速にシステムを回復させることができ、回復後も高速に処理を実行する事ができる。

【0147】また、上記マネージャに代えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記第1の計算機に障害が発生した場合に、上記第2の計算機に上記第2のパッケージプログラムを運転させるとともに、上記複数の計算機のうちの第3の計算機に上記第1のパッケージプログラムを起動状態で待機させるマネージャを備えたため、高速にシステムを回復させることができ、回復後も高速に処理を実行することができる。

【0148】また、上記複数の計算機のそれぞれで起動されるパッケージプログラムのそれぞれの優先順位を記憶する管理テーブルを備え、上記マネージャに代えて、上記複数の計算機のそれぞれのクラスタデーモンから監視

視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記第1の計算機に障害が発生した場合に、上記管理テーブルから上記第1の計算機で運転されていた上記パッケージプログラムよりも優先順位の低いパッケージプログラムを検索し、この優先順位の低いパッケージプログラムの運転を停止させるとともに、上記優先順位の低いパッケージプログラムを運転していた計算機で上記第1の計算機で運転されていたパッケージプログラムを起動させるマネージャを備えたため、計算機がダウンした後も、重要な処理を優先的に運転することができる。

【0149】また、クラスタシステムを構成する複数の計算機のうちの1つの計算機に障害が発生した場合に、上記障害が発生した計算機で運転中のアプリケーションや各種のサービスを提供するパッケージプログラムを他の計算機で運転させるクラスタシステムにおいて、上記複数の計算機はそれぞれ、自己の計算機の障害及び回復を監視するとともに、上記パッケージプログラムの起動及び運転を制御するクラスタデーモンを備え、上記複数の計算機のうち1つの計算機は、上記クラスタデーモンに加えて、上記複数の計算機のそれぞれで起動されるパッケージプログラムのそれぞれの優先順位及び上記複数の計算機のそれぞれの負荷を記憶する管理テーブルと、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記複数の計算機のうちの第1の計算機の負荷があらかじめ定められた負荷よりも大きくなった場合に、上記管理テーブルを参照し、上記第1の計算機で運転しているパッケージプログラムのうちの優先順位の低いパッケージプログラムの運転を停止させ、停止させたパッケージプログラムを上記複数の計算機のうちの負荷があらかじめ定められた負荷よりも小さい計算機で起動させるマネージャと、を備えたため、負荷を自動的に分散し、重要な処理のレスポンスの劣化を防ぐことができる。

【0150】また、上記管理テーブルに代えて、上記複数の計算機上で起動されるパッケージプログラムのそれぞれの優先順位を記憶する管理テーブルを備え、上記クラスタデーモンは、自己の計算機のリソースを監視し、上記マネージャに代えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記クラスタデーモンにより監視されたリソースに変化が生じた場合に、上記優先順位に基づいて、上記複数の計算機のそれぞれで運転されているパッケージプログラムのそれぞれにリソースを割り当て直すマネージャを備えたため、重要な処理のレスポンスの劣化を防ぐことができ、あまり重要でない処理の継続もすることができる。

【0151】また、複数の計算機のそれぞれによって並列に運転される複数の上記パッケージプログラムを1つ

のグループとするグループ名を記憶する管理テーブルを備え、上記マネージャに代えて、上記複数の計算機のそれぞれのクラスタデーモンから監視の結果を受け取るとともに、上記クラスタデーモンを制御して、上記複数の計算機のうちの第1の計算機に障害が発生した場合に、上記管理テーブルから、上記複数の計算機のうちの計算機であって上記第1の計算機上で運転されていたパッケージプログラムと同じグループのパッケージプログラムを運転していない計算機を検索し、検索された計算機で上記第1の計算機が運転していたパッケージプログラムを起動させ、運転させるマネージャを備えたため、システムを回復させた後に、高速に並列処理を実行することができる。

【図面の簡単な説明】

【図1】 本発明の第1の実施形態におけるプロセス構成図である。

【図2】 本発明の第1の実施形態におけるマネージャの構成図である。

【図3】 本発明の第1の実施形態におけるユーザからの要求に対するマネージャの処理を示すフローチャート図である。

【図4】 本発明の第1の実施形態におけるマネージャのリソース状態監視の処理を示すフローチャート図である。

【図5】 本発明の第1の実施形態の運用例を示す図である。

【図6】 本発明の第2の実施形態におけるプロセス構成図である。

【図7】 本発明の第2の実施形態におけるマネージャの構成図である。

【図8】 本発明の第2の実施形態におけるエージェントの構成図である。

【図9】 本発明の第2の実施形態におけるエージェントの変形例を示す構成図である。

【図10】 本発明の第2の実施形態におけるマネージャのリソース状態監視の処理を示すフローチャート図である。

【図11】 本発明の第2の実施形態におけるエージェントのリソース状態監視の処理を示すフローチャート図である。

【図12】 本発明の第3の実施形態におけるプロセス構成図である。

【図13】 本発明の第3の実施形態の運用例を示す図である。

【図14】 本発明の第4の実施形態におけるプロセス構成図である。

【図15】 本発明の第4の実施形態の運用例を示す図である。

【図16】 本発明の第5の実施形態におけるマネージャの構成図である。

【図17】 本発明の第5の実施形態における設定ファイル例を示す図である。

【図18】 本発明の第6の実施形態における設定ファイル例を示す図である。

【図19】 本発明の第7の実施形態におけるマネージャの構成図である。

【図20】 本発明の第7の実施形態におけるモードの状態遷移図である。

【図21】 本発明の第7の実施形態の運用例を示す図である。

【図22】 本発明の第8の実施形態におけるマネージャの構成図である。

【図23】 本発明の第9の実施形態におけるマネージャの構成図である。

【図24】 本発明の第9の実施形態における管理テーブルの記憶内容構成例である。

【図25】 本発明の第9の実施形態におけるマネージャの処理を説明するフローチャートである。

【図26】 本発明の第9の実施形態におけるシステムの動作例を示す機能ブロック図である。

【図27】 本発明の第10の実施形態におけるマネージャの構成図である。

【図28】 本発明の第10の実施形態における管理テーブルの記憶内容構成例である。

【図29】 本発明の第10の実施形態におけるマネージャの処理を説明するフローチャートである。

【図30】 本発明の第10の実施形態におけるシステムの動作例を示す機能ブロック図である。

【図31】 本発明の第11の実施形態におけるマネージャの構成図である。

【図32】 本発明の第11の実施形態における管理テーブルの記憶内容構成例である。

【図33】 本発明の第11の実施形態におけるマネージャの処理を説明するフローチャートである。

【図34】 本発明の第11の実施形態におけるシステムの動作例を示す機能ブロック図である。

【図35】 本発明の第12の実施形態における管理テーブルの記憶内容構成例である。

【図36】 本発明の第12の実施形態におけるマネージャの処理を説明するフローチャートである。

【図37】 本発明の第12の実施形態におけるシステムの動作例を示す機能ブロック図である。

【図38】 本発明の第13の実施形態における管理テーブルの記憶内容構成例である。

【図39】 本発明の第13の実施形態におけるシステムの動作例を示す機能ブロック図である。

【図40】 本発明の第14の実施形態における管理テーブルの記憶内容構成例である。

【図41】 本発明の第14の実施形態におけるマネー

ジャの処理を説明するフローチャートである。

【図42】 本発明の第14の実施形態におけるシステムの動作例を示す機能ブロック図である。

【図43】 本発明の第15の実施形態における管理テーブルの記憶内容構成例である。

【図44】 本発明の第15の実施形態におけるマネージャの処理を説明するフローチャートである。

【図45】 本発明の第15の実施形態におけるシステムの動作例を示す機能ブロック図である。

10 【図46】 本発明の第15の実施形態におけるシステムの動作例を示す機能ブロック図である。

【図47】 本発明の第16の実施形態におけるマネージャの構成図である。

【図48】 本発明の第16の実施形態における管理テーブルの記憶内容構成例である。

【図49】 本発明の第16の実施形態におけるマネージャの処理を説明するフローチャートである。

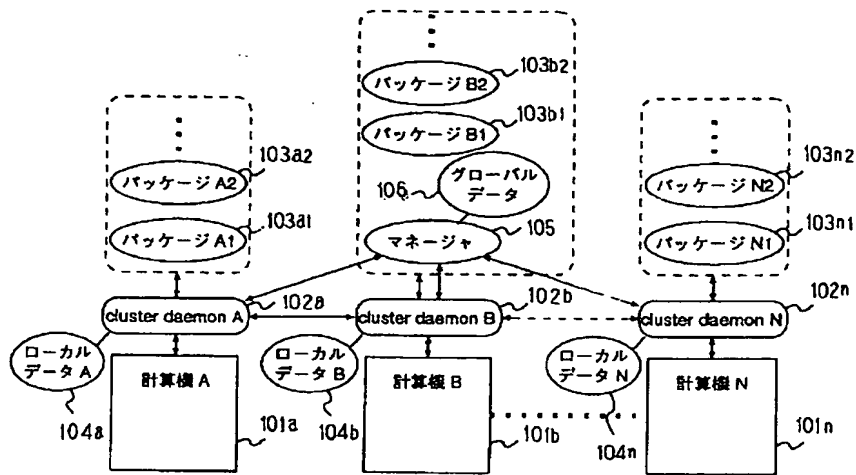
【図50】 従来技術を示すプロセス構成図である。

【図51】 従来技術の動作を示す図である。

20 【符号の説明】

101a 計算機A、101b 計算機B、101n 計算機N、102a cluster daemon A、102b cluster daemon B、102n cluster daemon N、103a1 パッケージA1、103b1 パッケージB1、103n1 パッケージN1、103a2 パッケージA2、103b2 パッケージB2、103n2 パッケージN2、104a ローカルデータA、104b ローカルデータB、104n ローカルデータN、105 マネージャ、106 グローバルデータ、201 要求処理機構、202 リソース制御機構、203 リソース状態DB、204 リソース状態監視機構、205 リソース状態変化処理機構、206 通知設定機構、207 通知設定DB、208 リソース状態変化通知機構、209 通信制御機構、501a プロセスA、501b プロセスB、501c プロセスC、601a エージェントA、601b エージェントB、601n エージェントN、801 リソース状態DB、1301 ユーザプロセスA、1302 ユーザプロセスB、1501 ユーザプロセスA、1502 ユーザプロセスB、1601 設定ファイル、1602 自動制御機構、1901 モード管理機構、2201 ログ管理機構、2202 ログDB、2301 通知処理機構、2303 管理テーブル、2302 モード管理機構、2304 モード制御機構、2701 出力管理機構、2702 管理テーブル、2703 出力抑止機構、3101 パッケージ管理機構、3102 管理テーブル、3103 パッケージ制御機構。

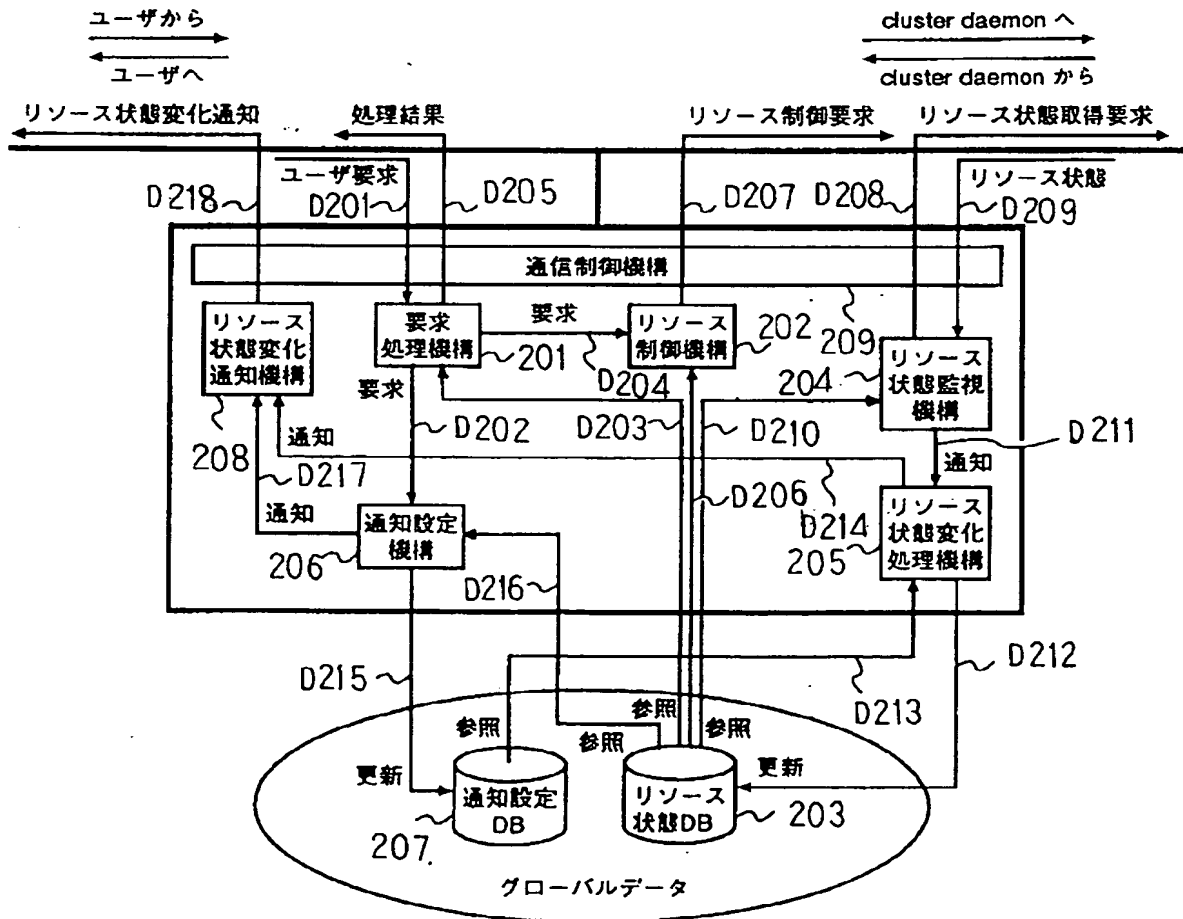
【図1】



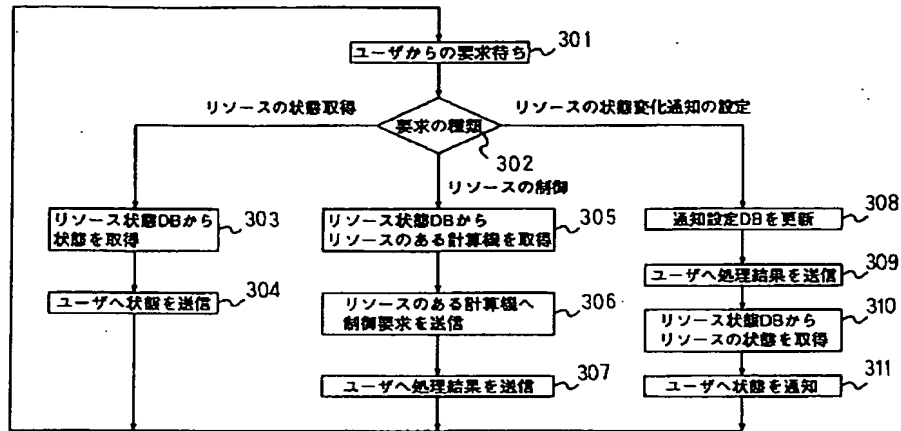
【図18】

- 1: # パッケージ名: 条件
 2: pkg1: pkg2
 3: pkg2: pkg1
 4: pkg3: pkg1pkg2

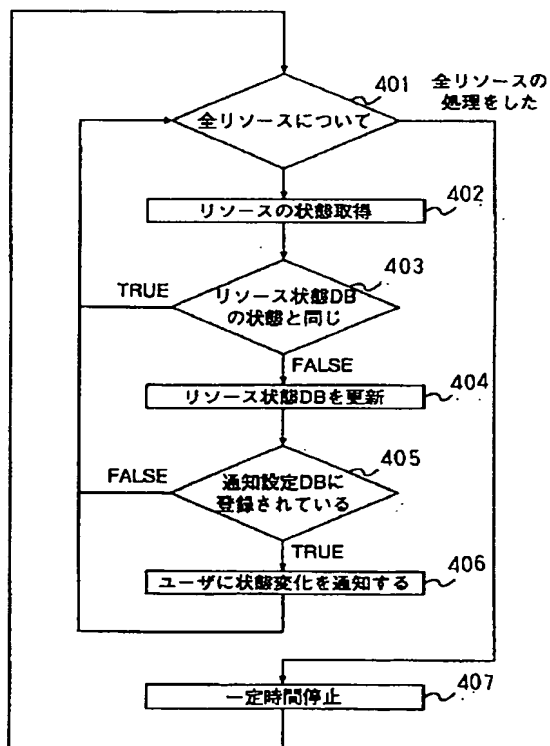
【図2】



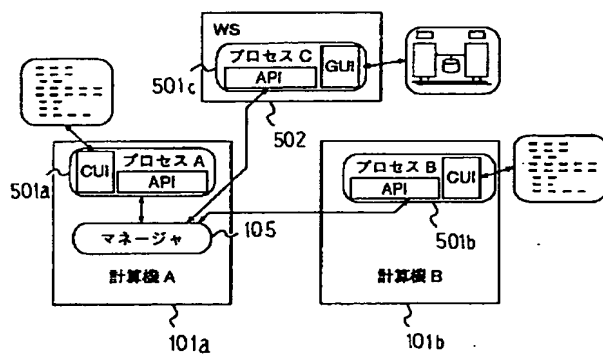
【図3】



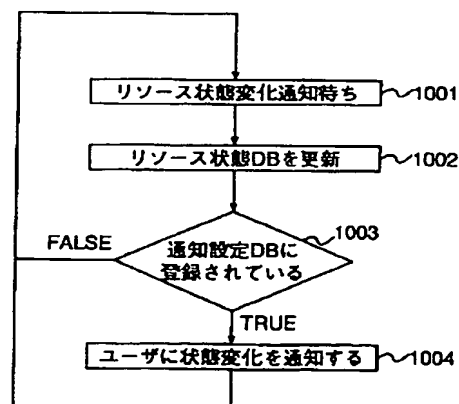
【図4】



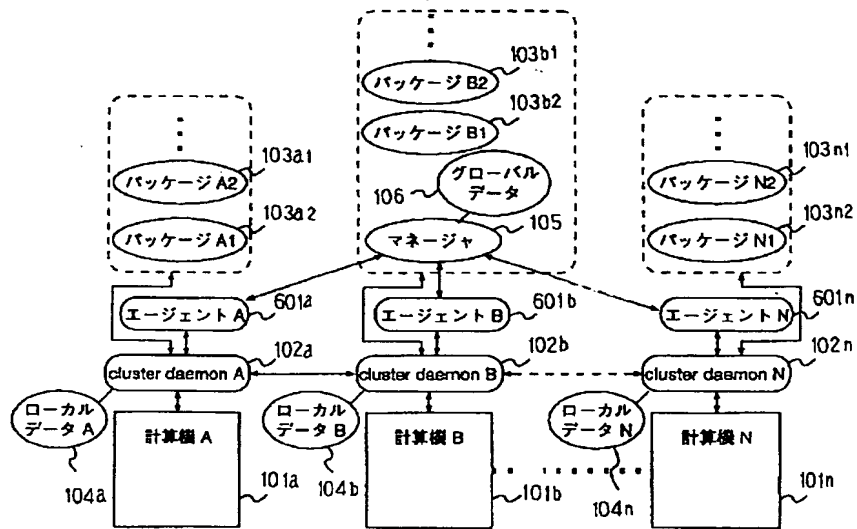
【図5】



【図10】



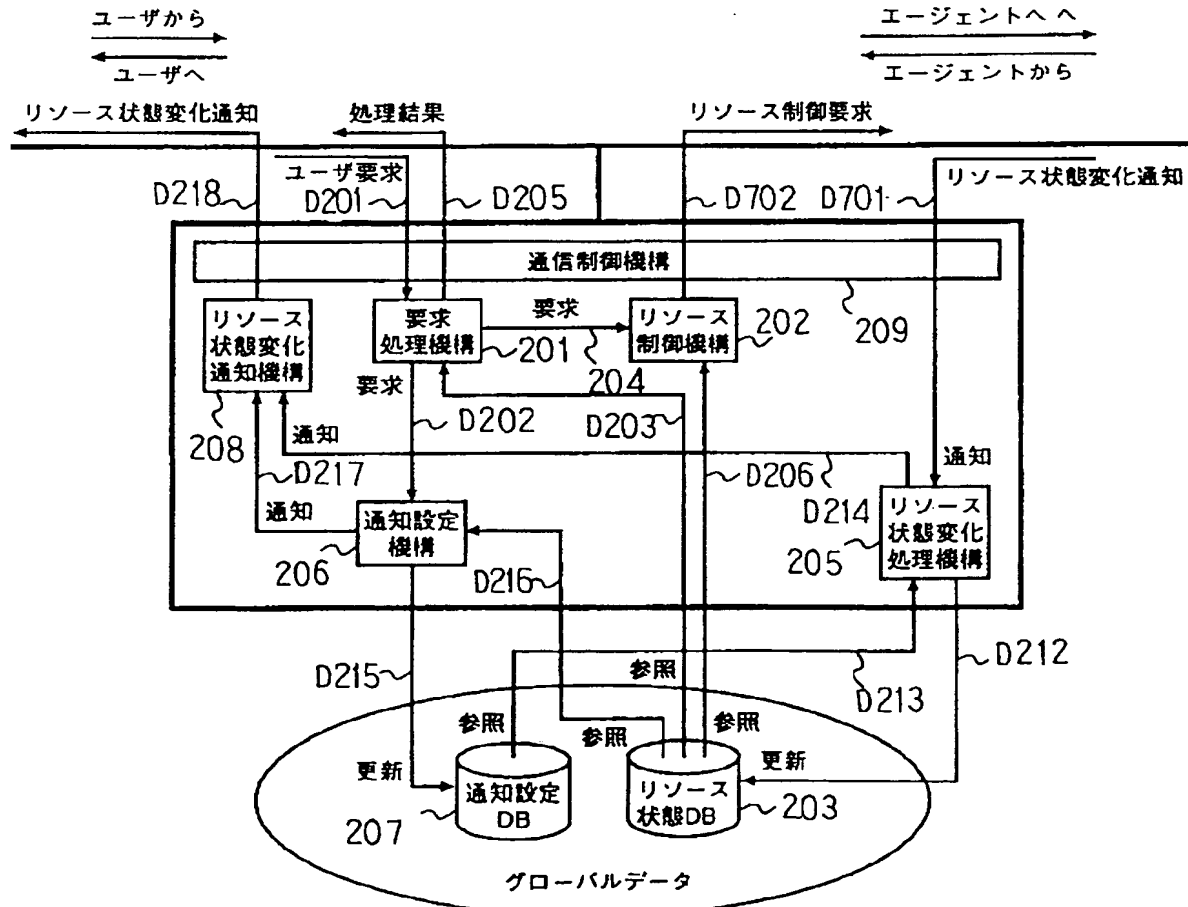
【図6】



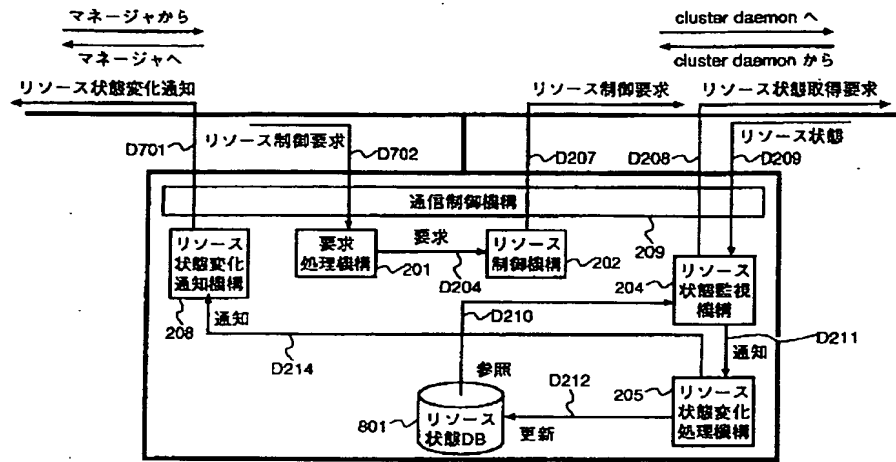
【図24】

パッケージ名	モード
pkgA	運転
pkgB	待機

【図7】



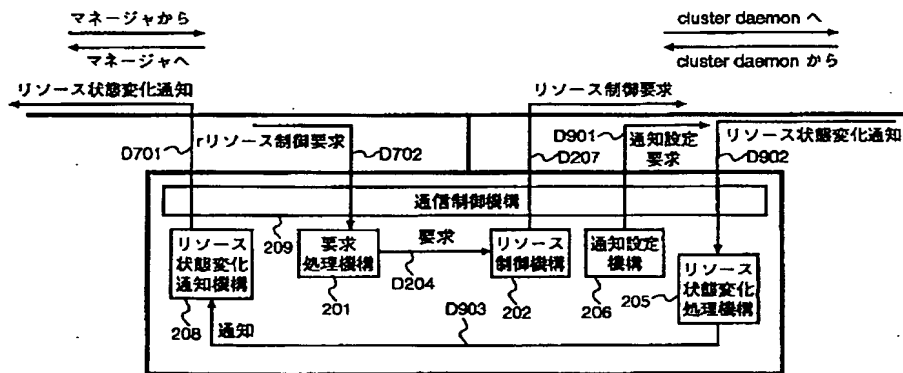
【図8】



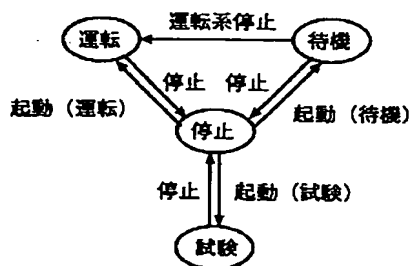
【図28】

パッケージ名	モード
pkgA	解除
pkgB	抑止

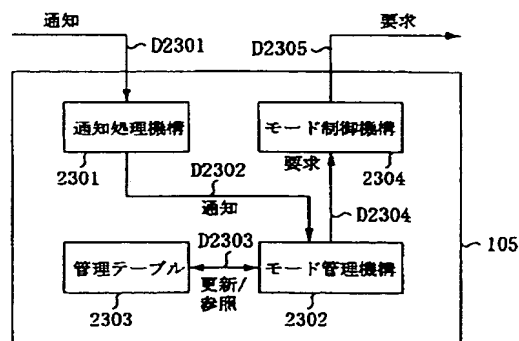
【図9】



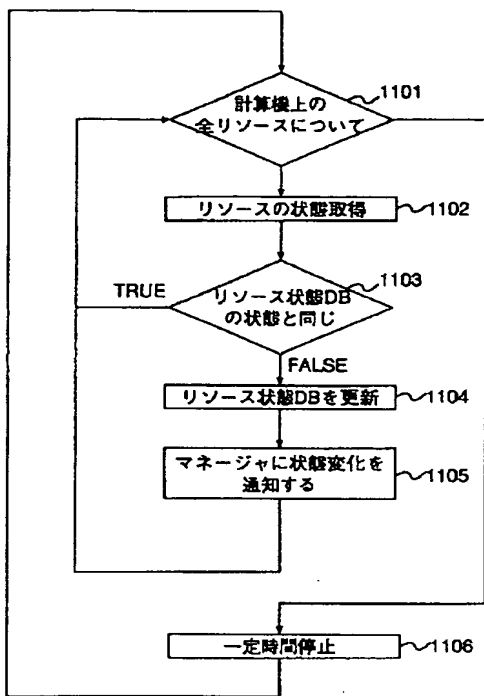
【図20】



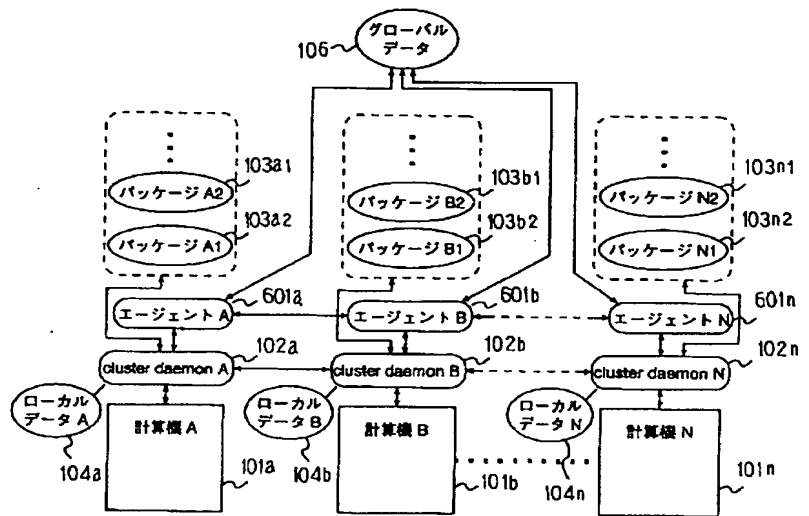
【図23】



【図11】

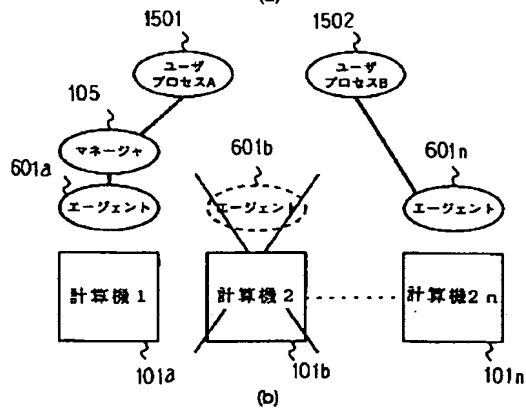
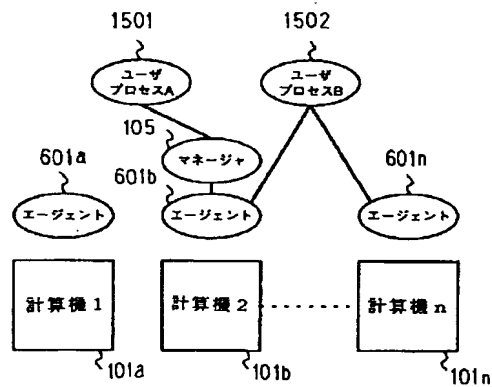
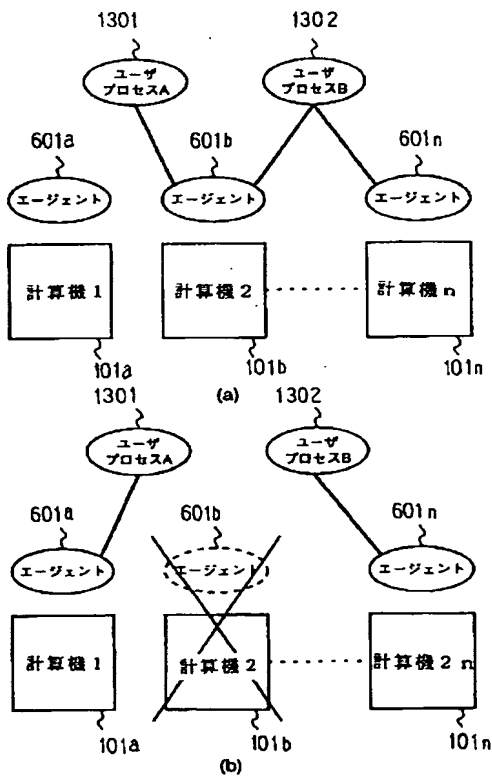


【図12】

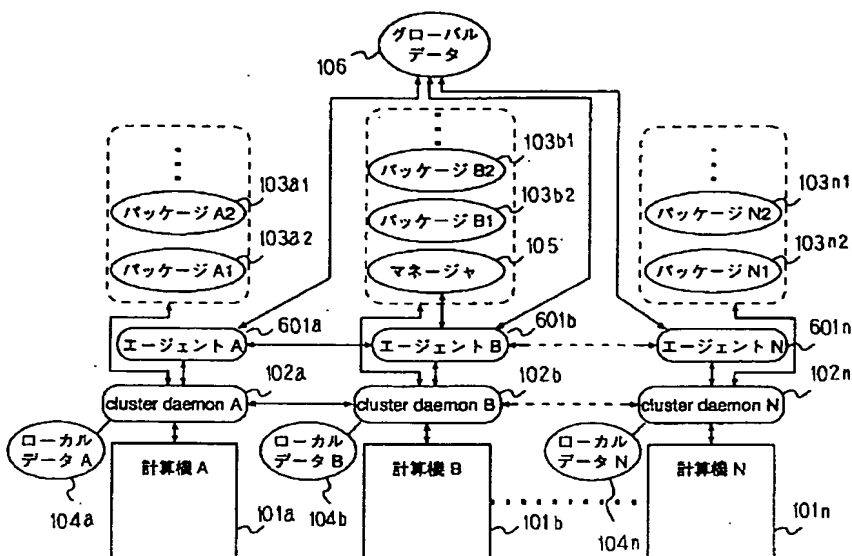


【図15】

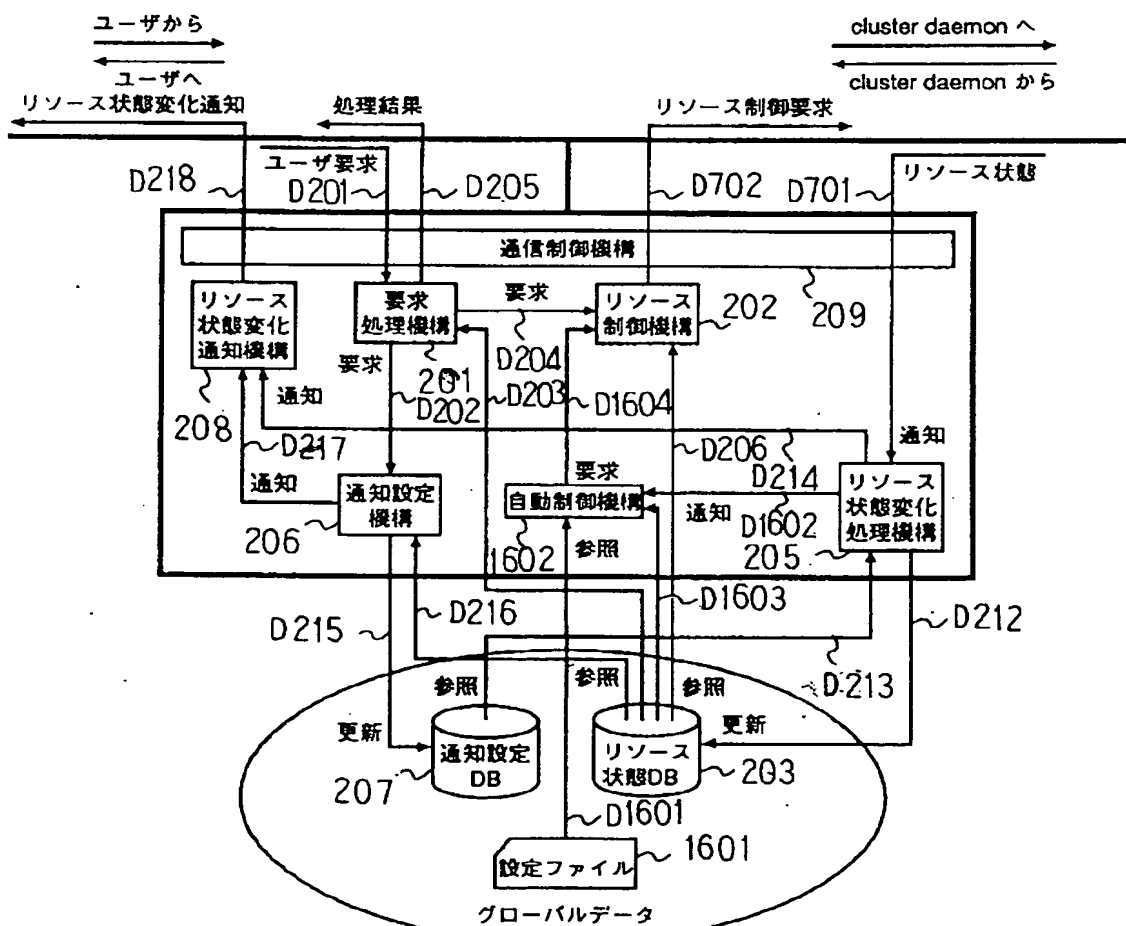
【図13】



【図14】



【図16】



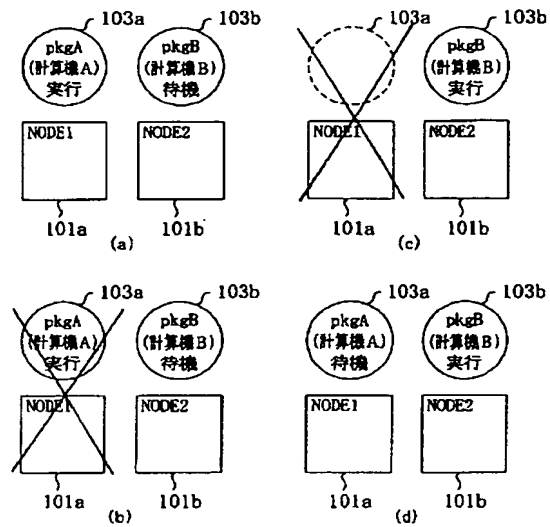
【図17】

```

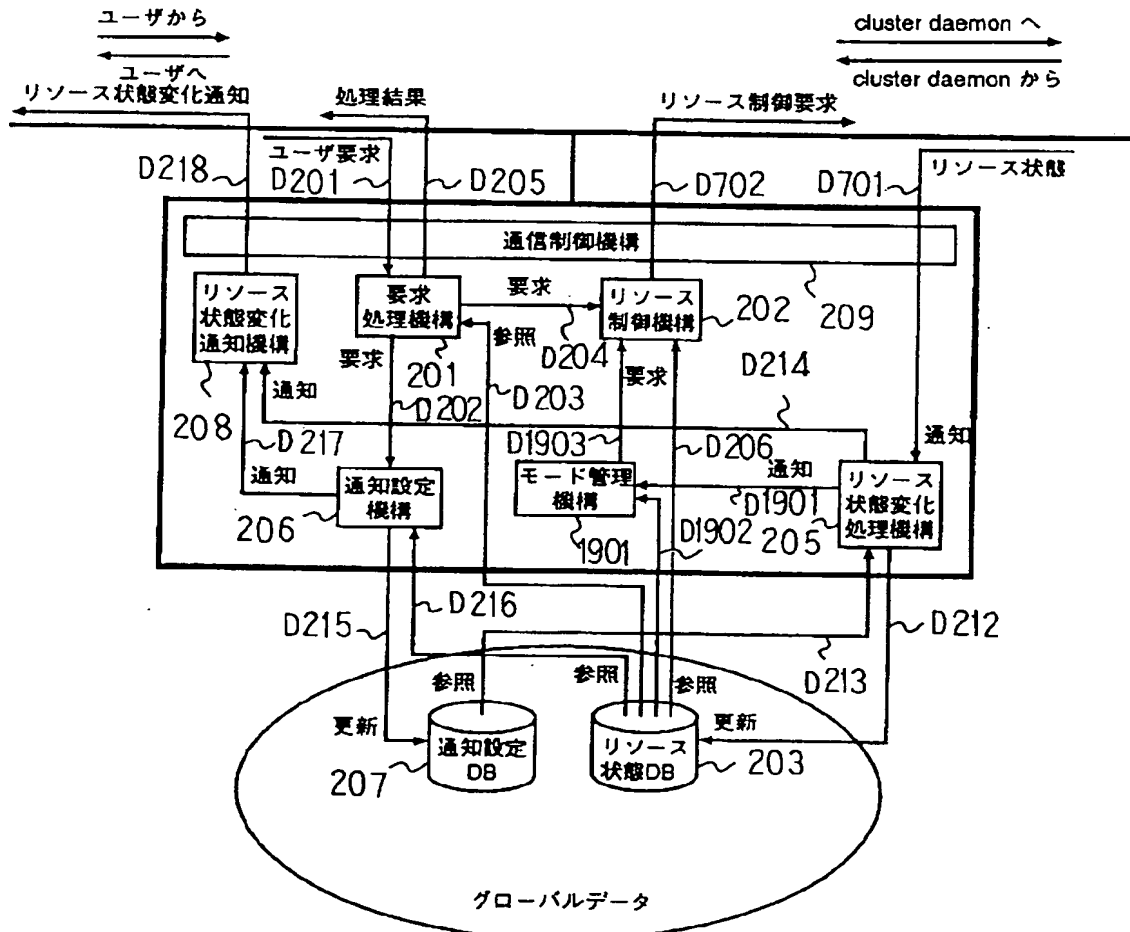
1: # リソース名 (イベント名) {
2: #     処理
3: # }
4:
5: pkg1 (halt) {
6:     if (pkg2.state == down){
7:         pkg2->startup
8:     }
9: }
10:
11: pkg1 (startup) {
12:     if (pkg2.state == up && pkg2.node == pkg1.node){
13:         pkg2->halt
14:     }
15: }
16:
17: pkg2 (startup) {
18:     if (pkg1.state == up && pkg2.node == pkg1.node){
19:         pkg2->halt
20:     }
21: }

```

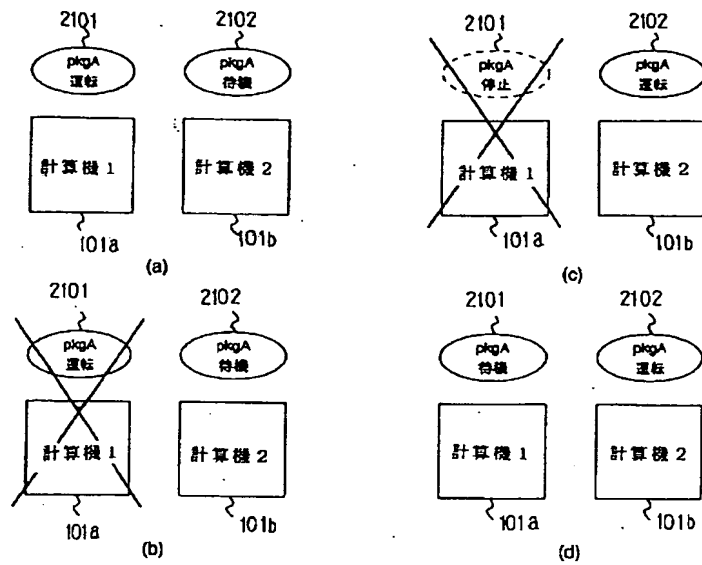
【図26】



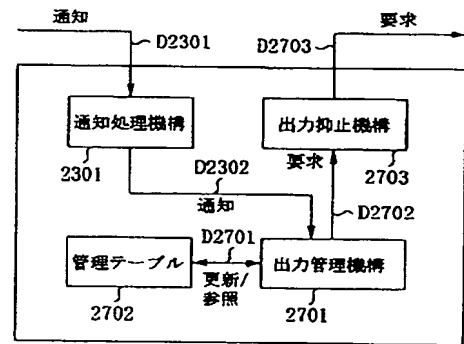
【図19】



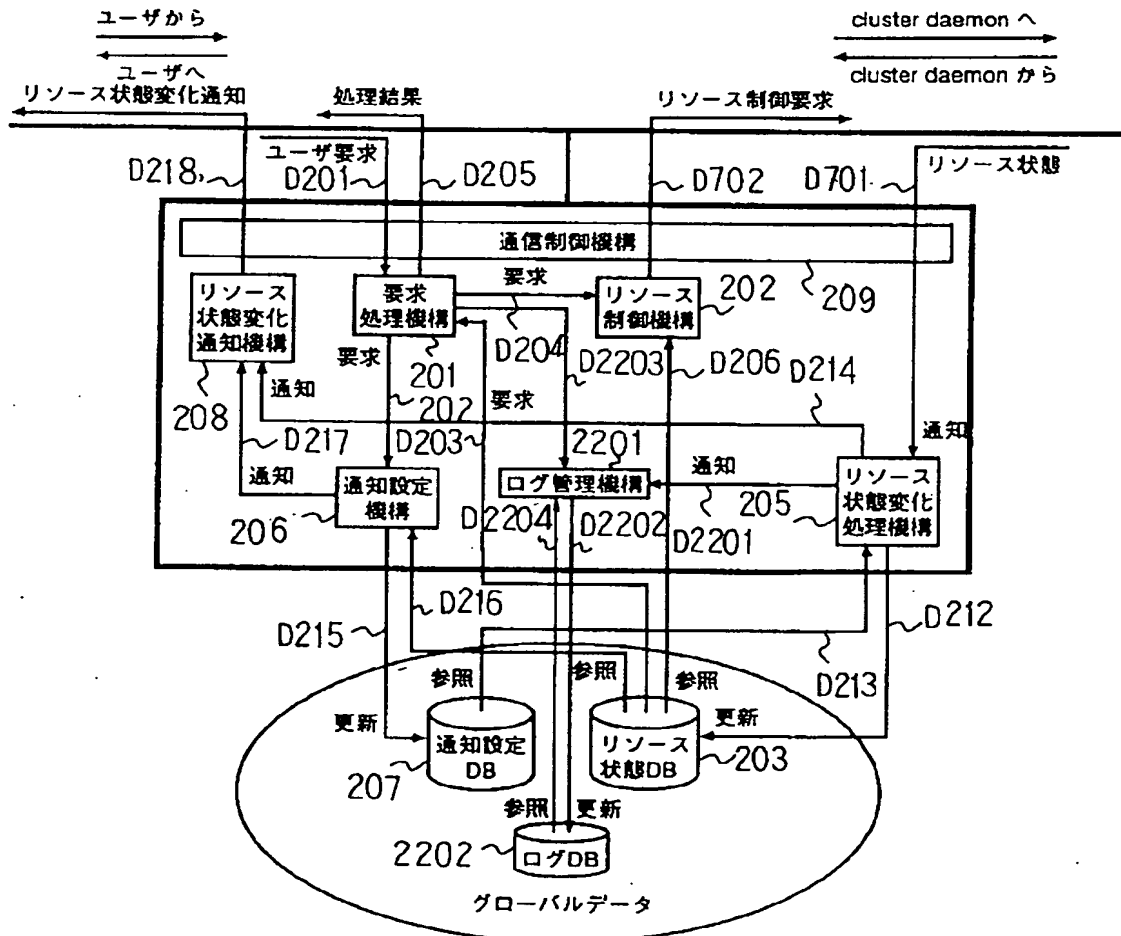
【図21】



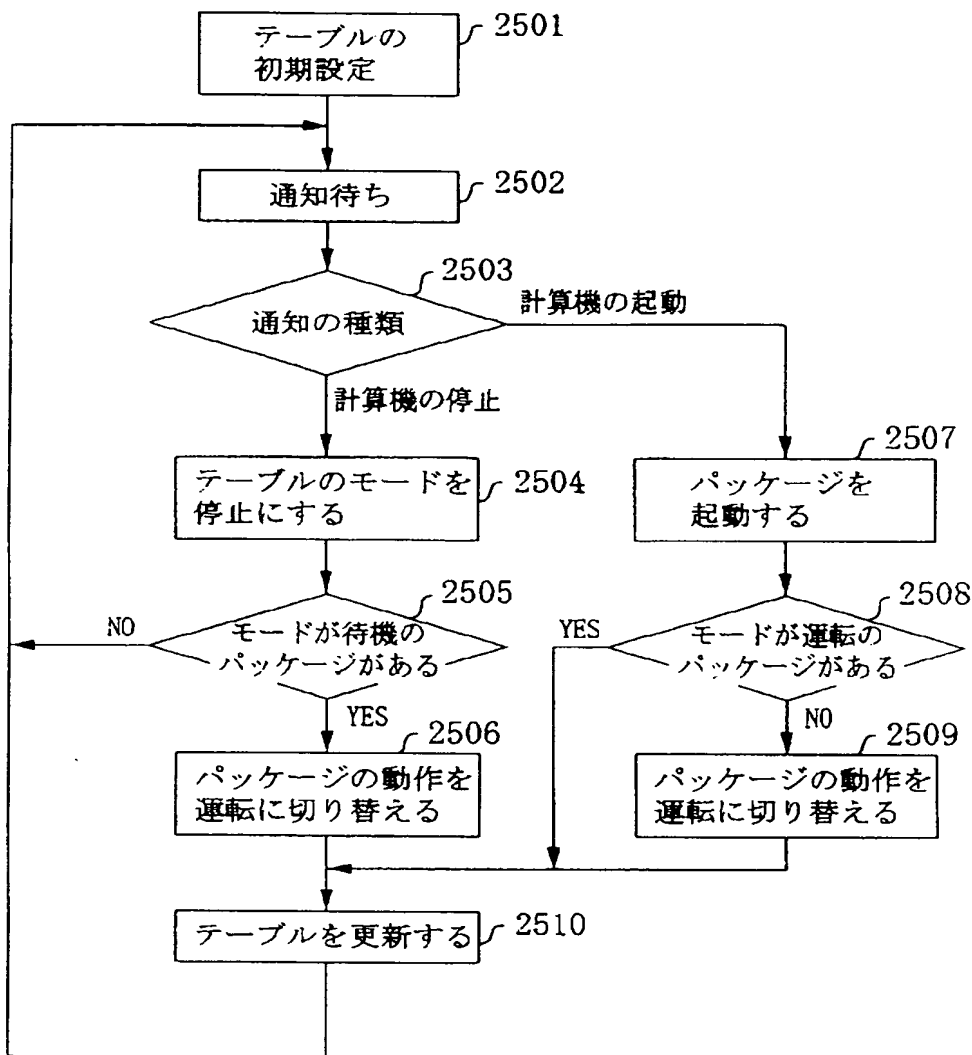
【図27】



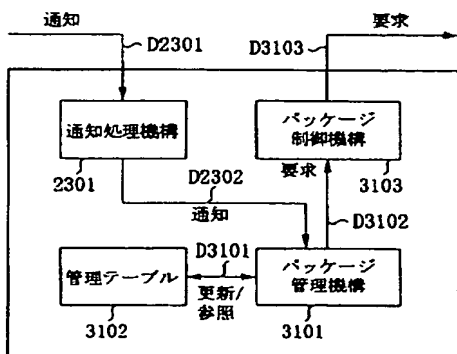
【図22】



【図25】



【図31】



【図32】

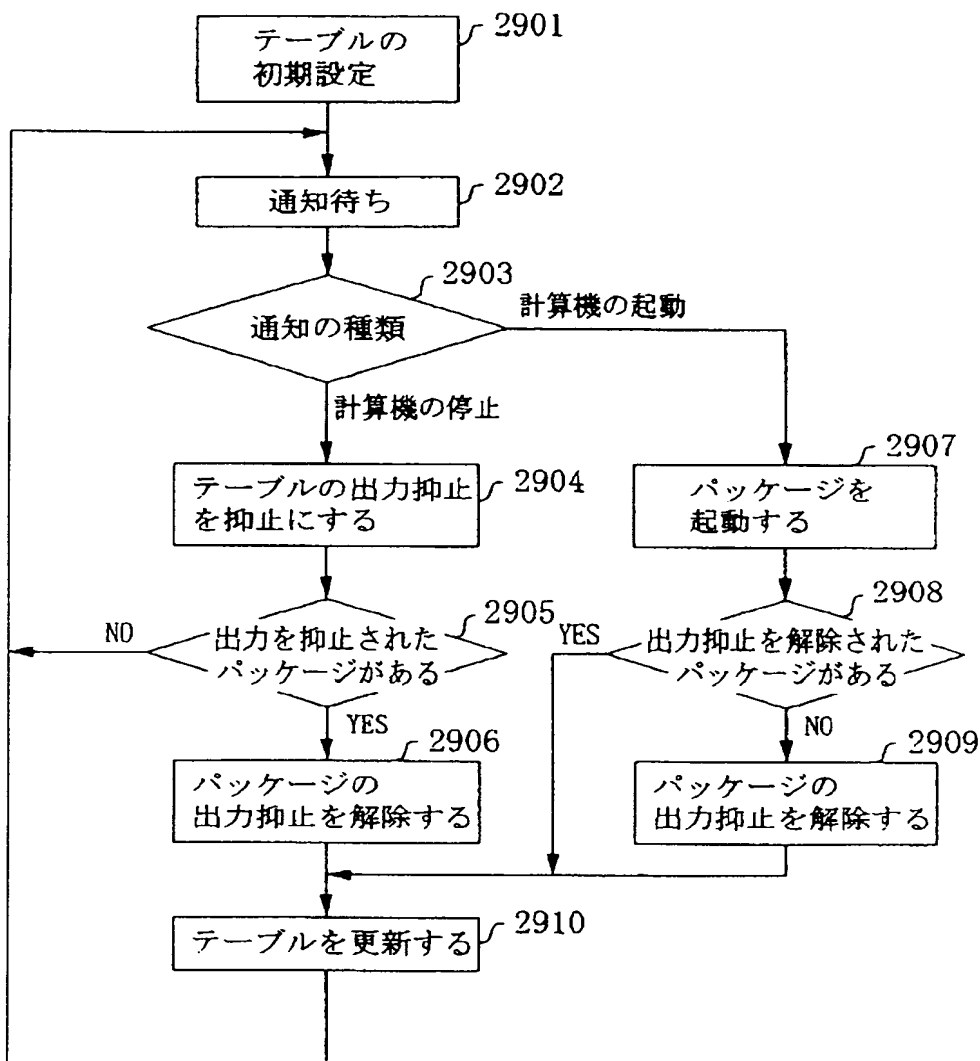
パッケージ名	モード	実行計算機	グループ名
pkgA	運転	NODEn	group1
pkgB	待機	NODE2	group1
pkgC	運転	NODE3	group2
pkgD	待機	NODE2	group2

(a)

計算機	状態
NODE1	停止
NODE2	運転
NODE3	運転
NODE4	運転
NODEn	運転

(b)

【図29】



【図35】

パッケージ名	実行計算機
pkgA	NODE1
pkgB	NODE2
pkg(n-1)	NODE(n-1)
空き	NODEn

(a)

計算機	状態
NODE1	停止
NODE2	運転
NODEn	運転

(b)

【図38】

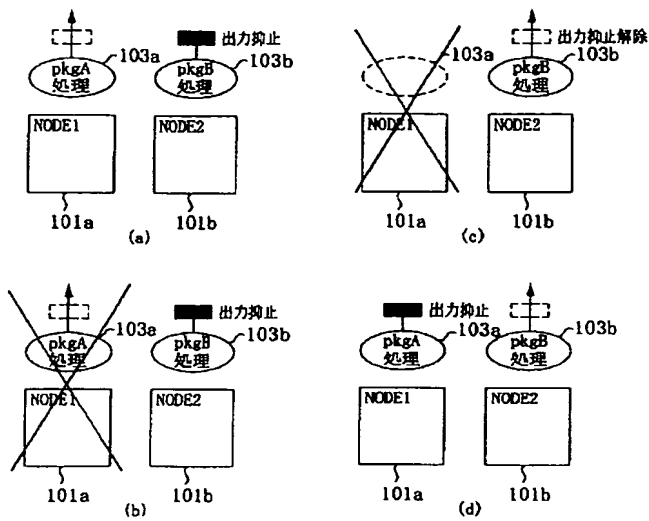
パッケージ名	実行計算機	グループ名
pkgA	NODE1	group1
pkgB	NODE2	group1
pkgB	NODE1	group2

(a)

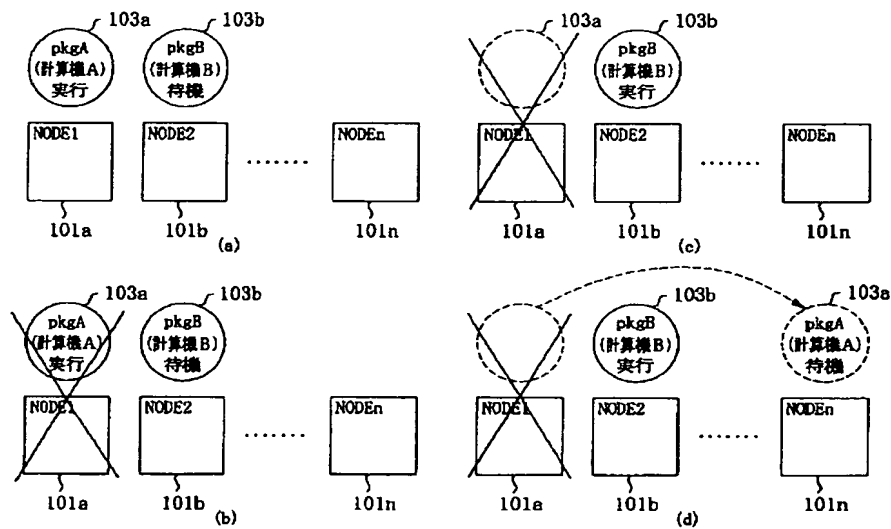
計算機	状態
NODE1	運転
NODE2	運転
NODEn	運転

(b)

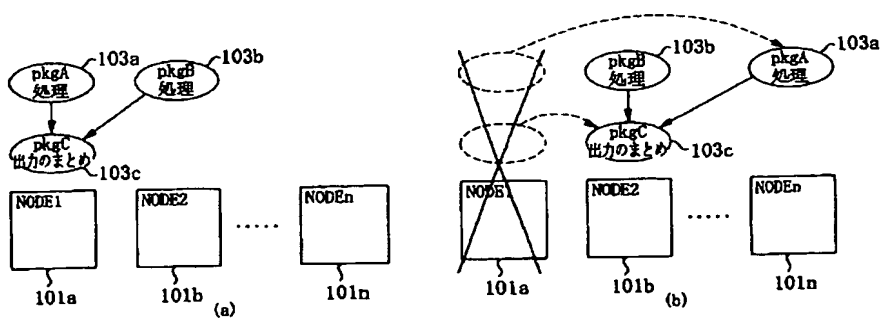
【図30】



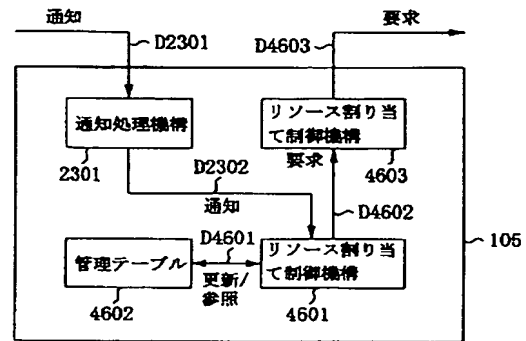
【図34】



【図39】



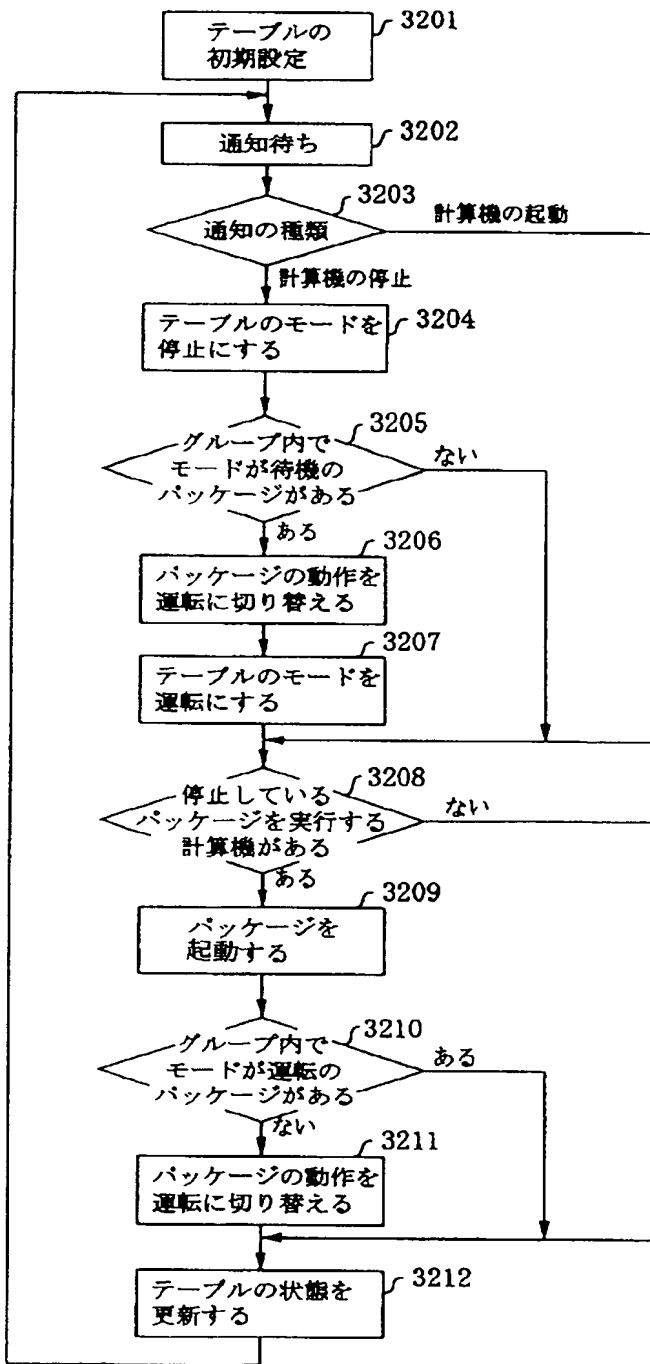
【図47】



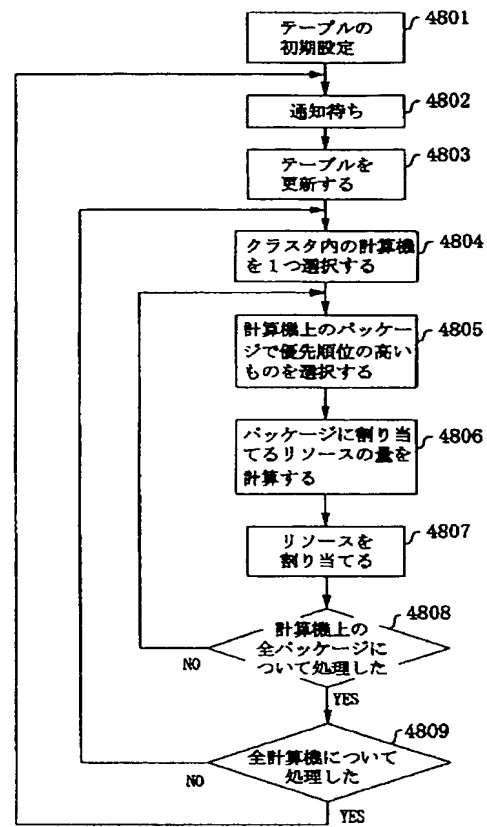
【図48】

パッケージ名	実行計算機	優先順位
pkgA	NODE1	1
pkgB	NODE2	2
pkgN	NODEn	n

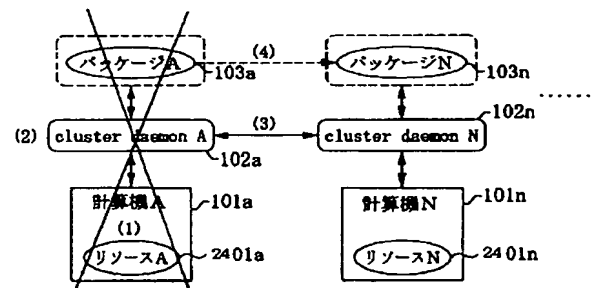
【図33】



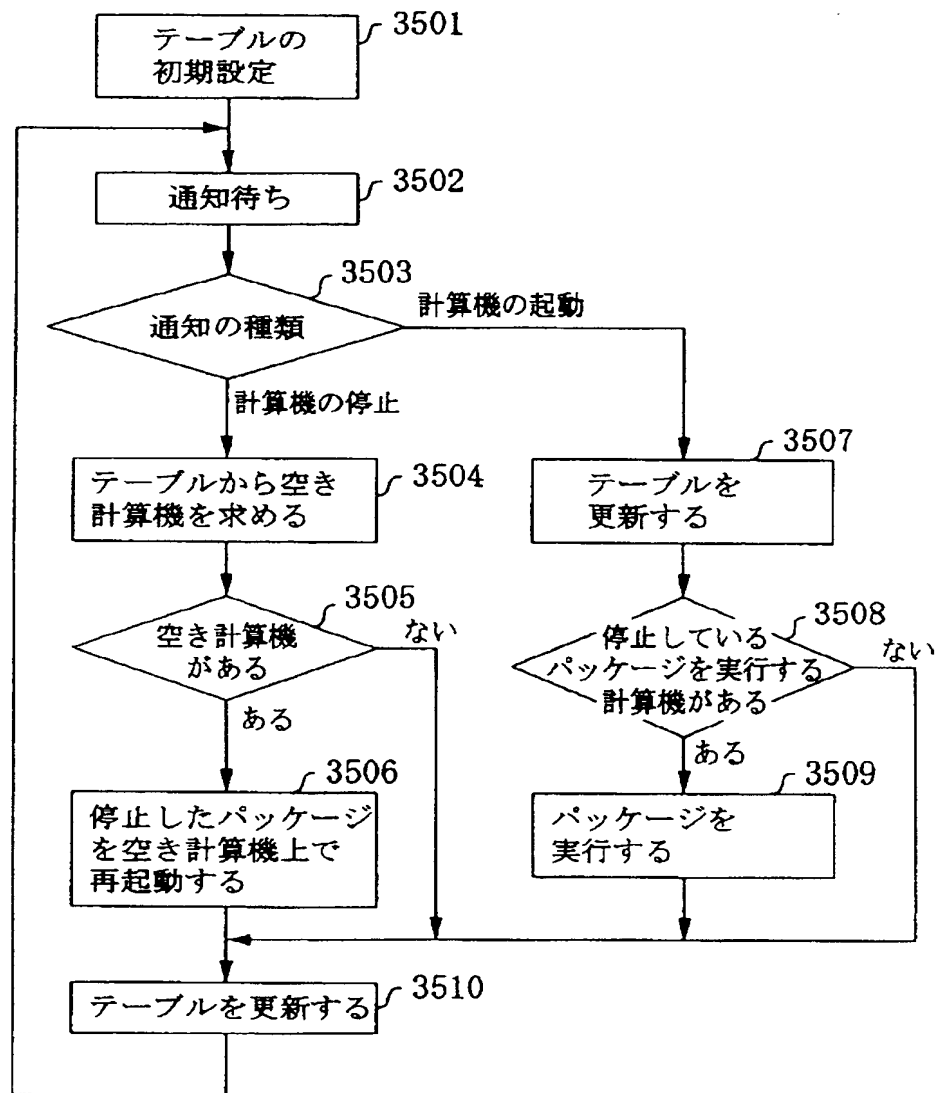
【図49】



【図51】



【図36】



【図40】

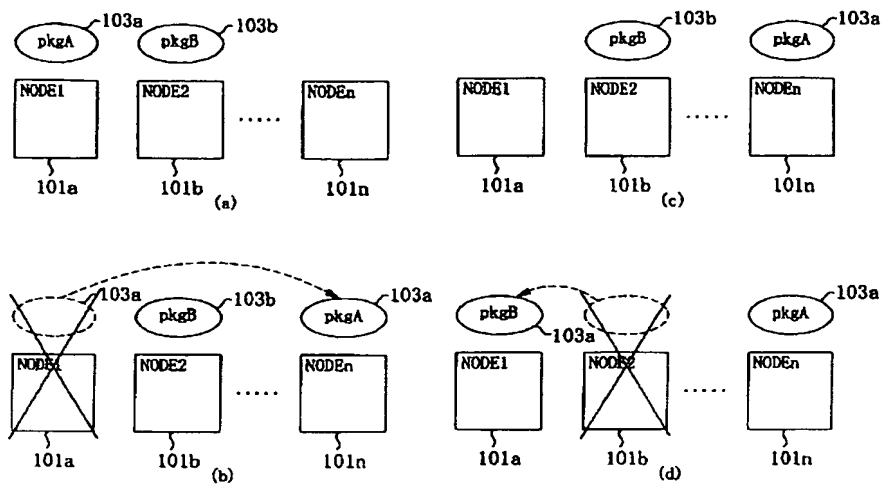
パッケージ名	実行計算機	優先順位
pkgA	NODE1	1
pkgB	NODE2	2
pkgN	NODEn	n

(a)

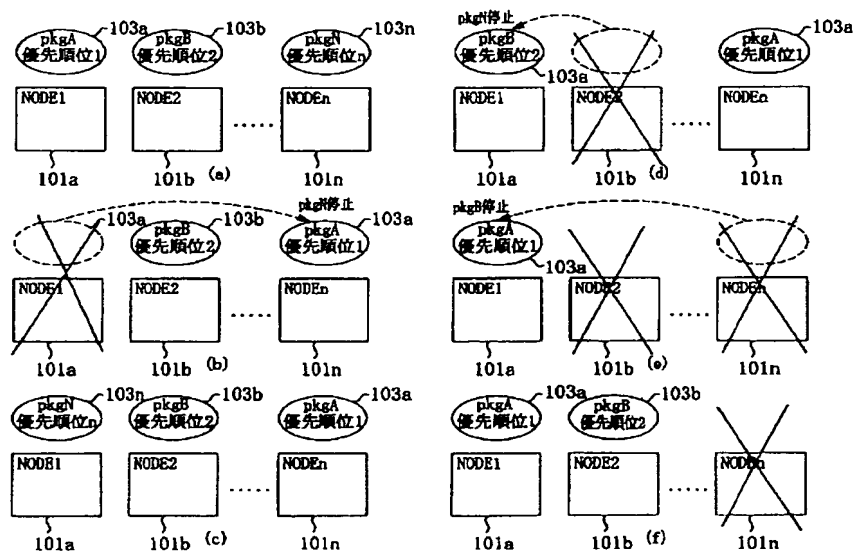
計算機	状態
NODE1	運転
NODE2	運転
NODEn	運転

(b)

【図37】



【図42】



【図43】

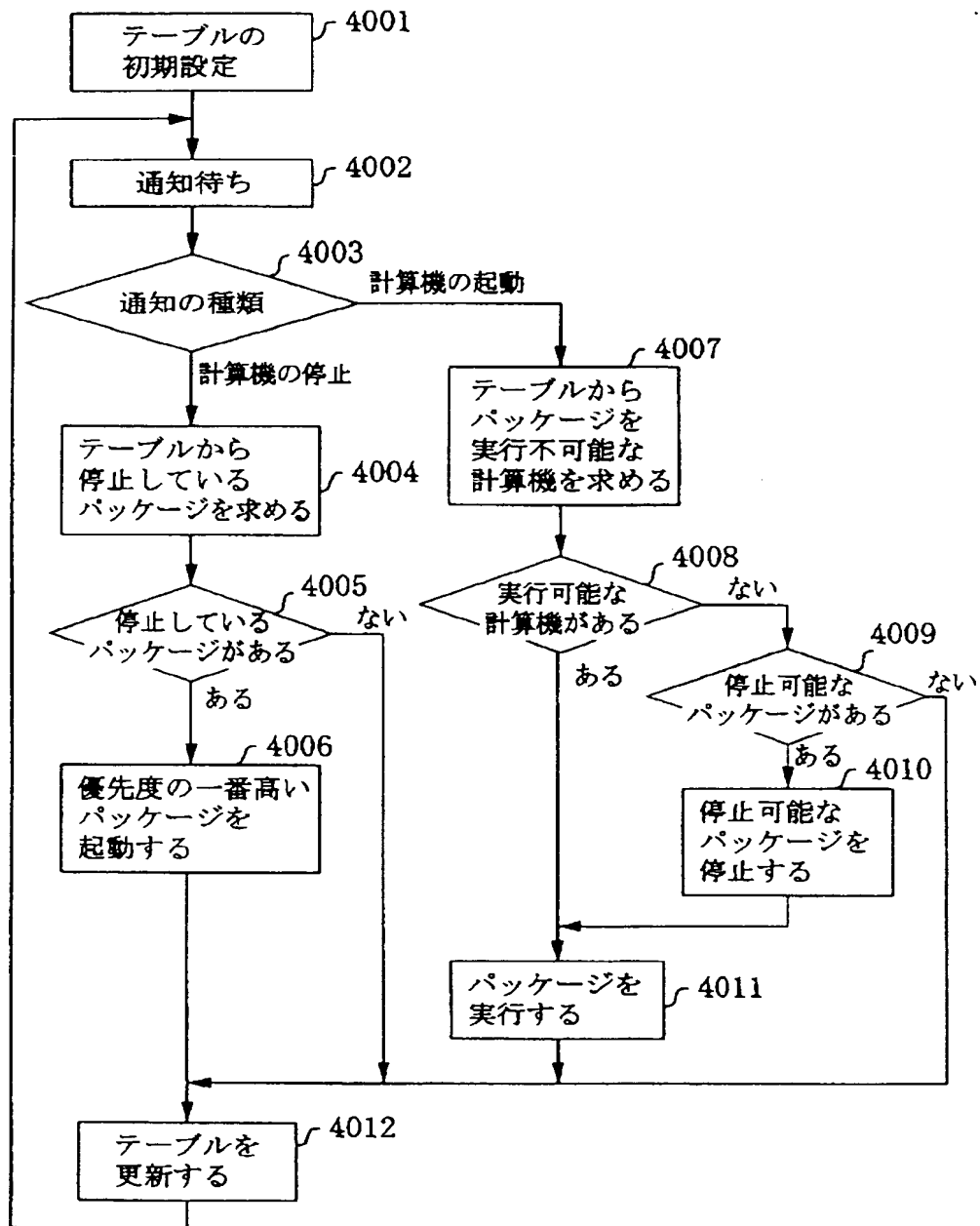
パッケージ名	実行計算機	優先順位	負荷
pkgA	NODE1	1	40
pkgB	NODE2	2	40
pkgC	NODE3	3	40
pkgD	NODE1	4	20
pkgE	NODE2	5	20
pkgF	NODE3	6	20

(a)

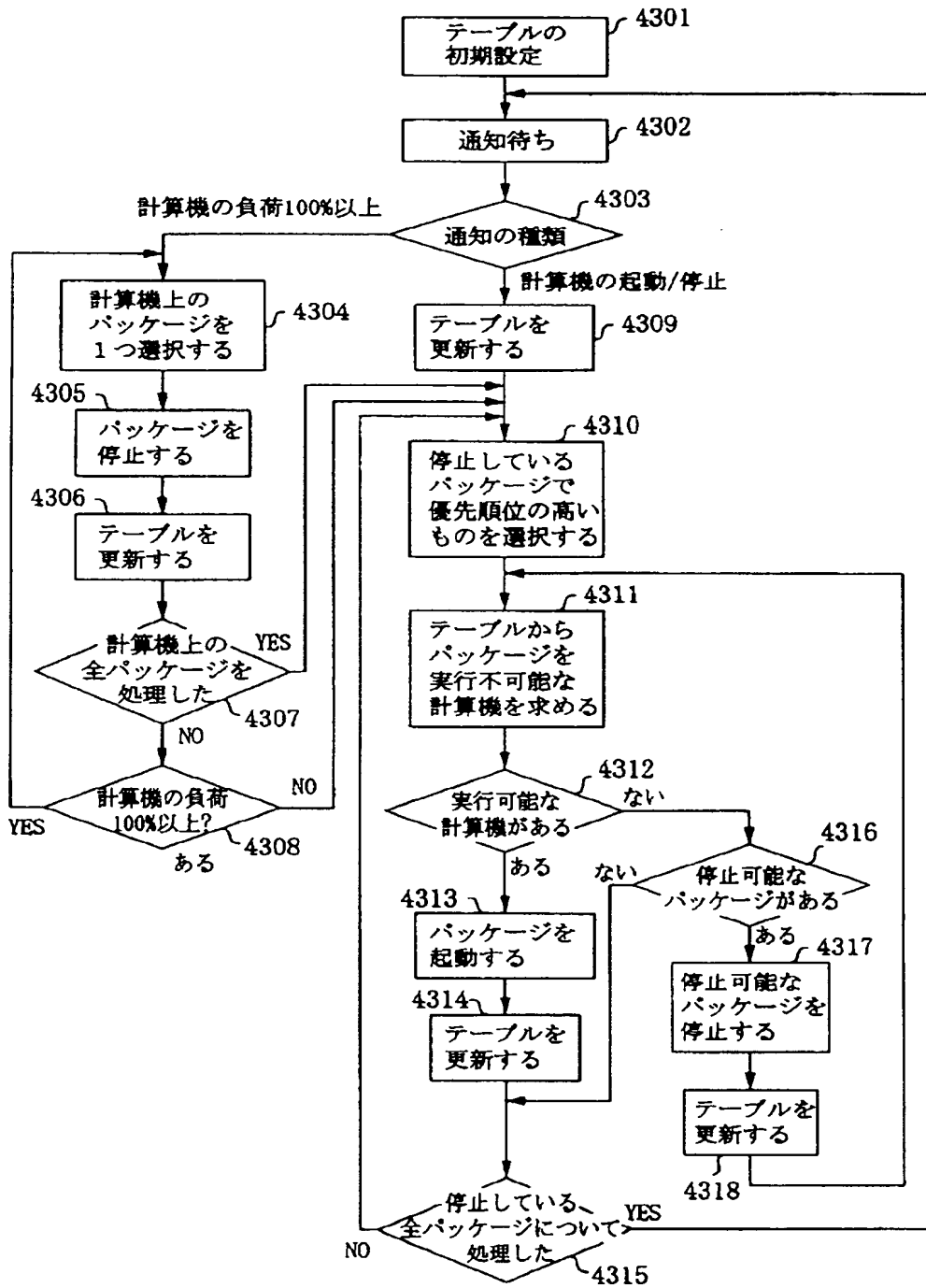
計算機	状態	負荷
NODE1	運転	60
NODE2	運転	60
NODEn	運転	60

(b)

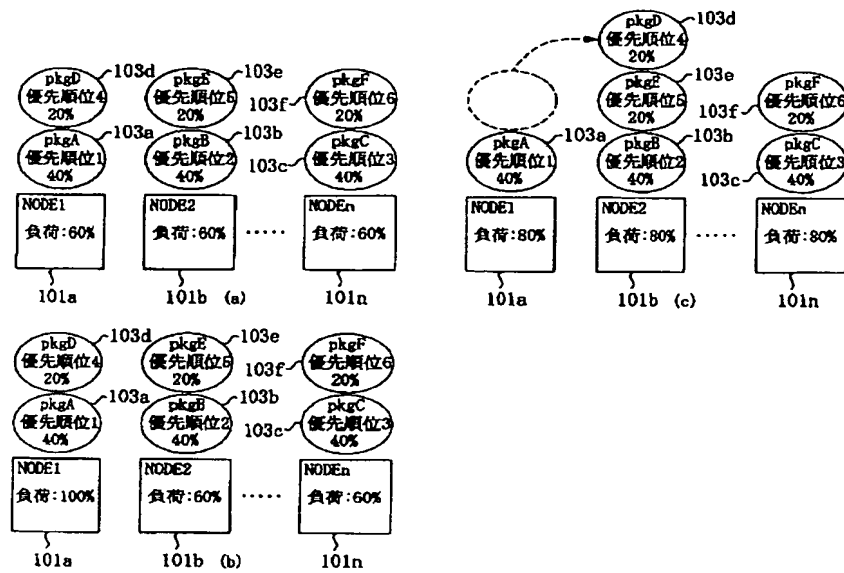
【図41】



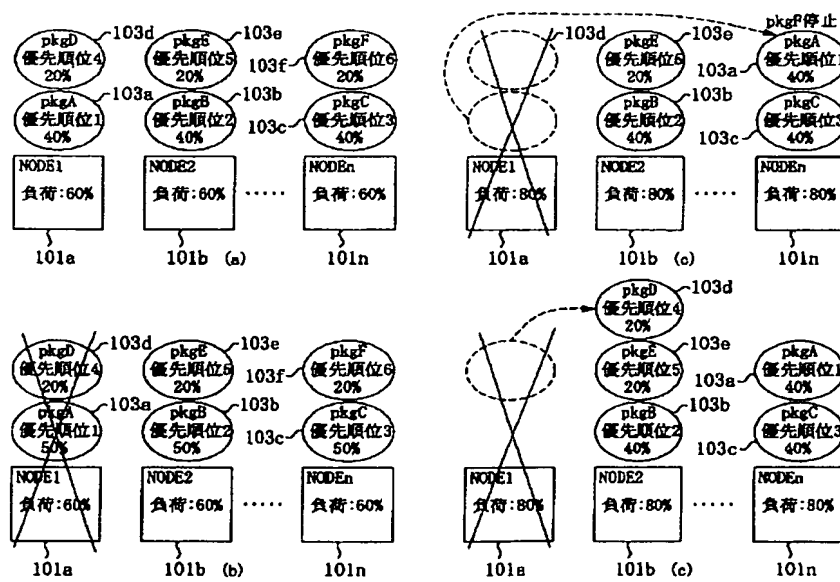
【図44】



【図45】



【図46】



【図50】

